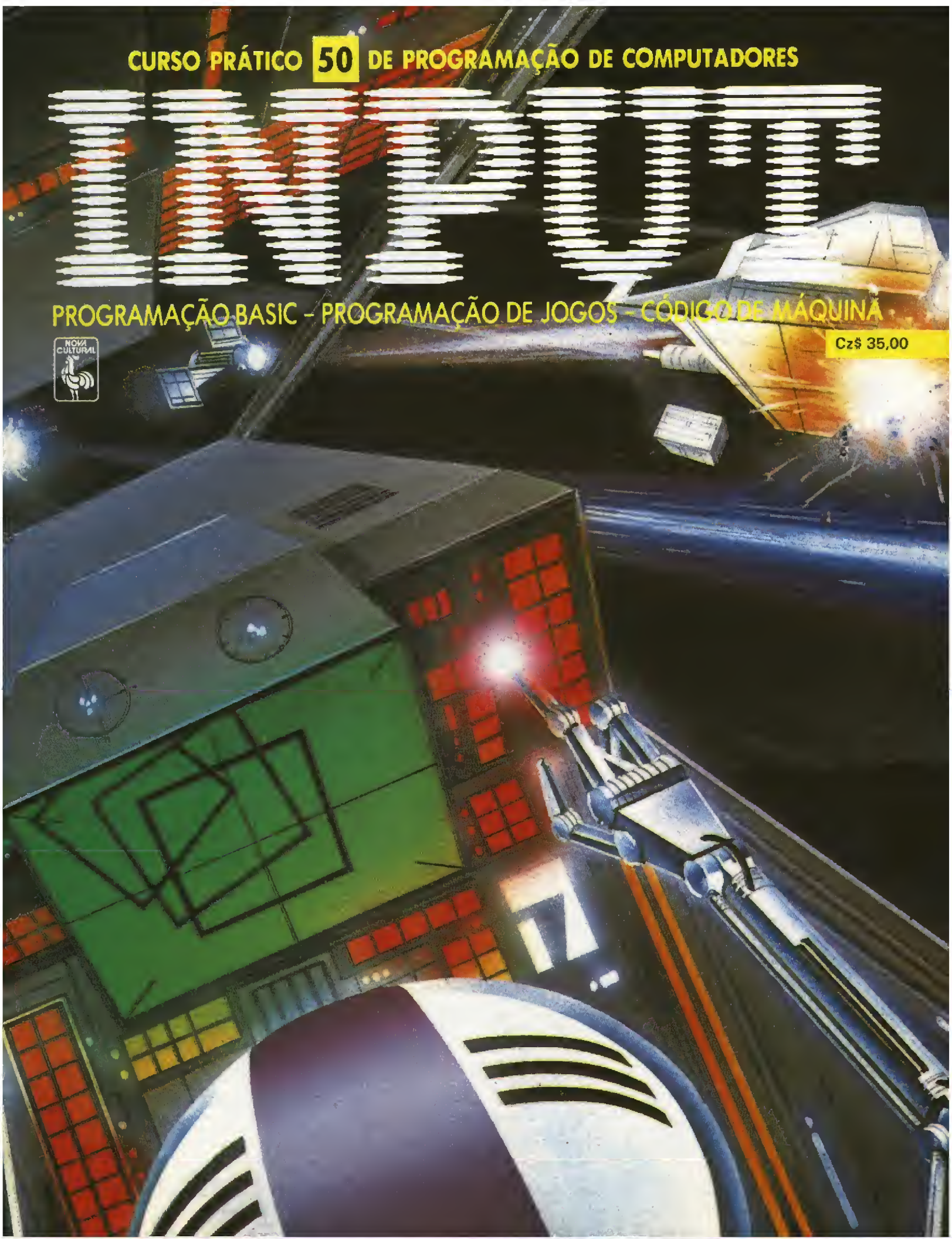
CURSO PRÁTICO 50 DE PROGRAMAÇÃO DE COMPUTADORES

TRS-80

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

NOVA CULTURAL

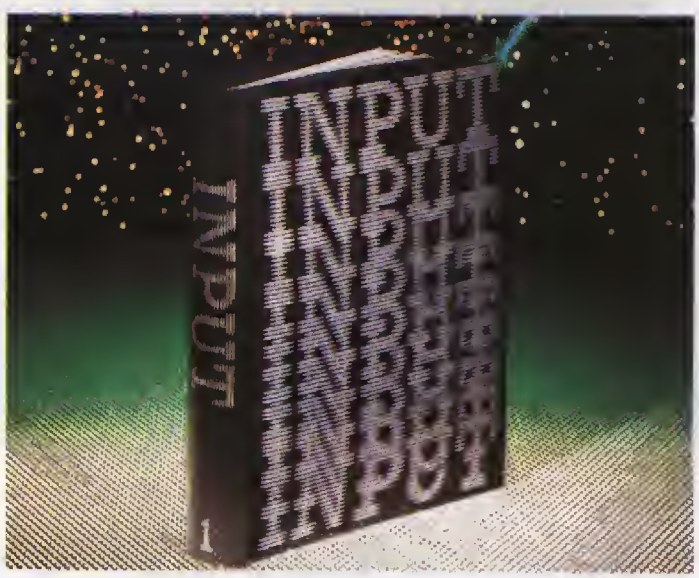
Cz$ 35,00



INPUT

Vol. 4 N.º 50

NESTE NÚMERO



PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: 1. PESSOALMENTE — Por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em **São Paulo**, os endereços são: rua Brigadeiro Tobias, 773, Centro; avenida Industrial, 117, Santo André; e no **Rio de Janeiro**: avenida Mem de Sá, 191/193, Centro. 2. POR CARTA — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidora Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132, Jardim Teresa — CEP 06000 — Osasco — SP. Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na agência do Correio. 3. POR TELEX — Utilize o nº (011) 33 670 DNAP.

Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações, Lda. — Qta. Pau Varais, Azinhaga de Fetais — 2 685, Camarate — Lisboa; Apartado 57 — Telex 43 069 JARLIS P.

Atenção: Após seis meses do encerramento da coleção, os pedidos serão atendidos dependendo da disponibilidade do estoque.

Obs.: Quando pedir livros, mencione sempre título e/ou autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

Editor
VICTOR CIVITA

REDAÇÃO
Diretor Editorial: Carmo Chagas

Editores Executivos: Antonio José Filho, Berta Sztark Amar

Editor Chefe: Paulo de Almeida
Editor de Texto: Cláudio A. V. Cavalcanti
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Dagmar Bastos Sampaio, Grace Alonso Arruda, Monica Lenardon Corradi
Secretária de Redação/ Coordenadora: Stefania Crema
Secretários de Redação: Beatriz Hagström, José Benedito de Oliveira Damião, Maria de Lourdes Carvalho, Marisa Soares de Andrade, Mauro de Queiroz

COLABORADORES

Consultor Editorial Responsável: Dr. Renato M. E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda., Campinas, SP
Tradução, adaptação, programação e redação: Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Marcos Huascar Velasco, Raul Neder Porrelli, Ricardo J. P. de Aquino Pereira
Coordenação Geral: Rejane Felizatti Sabbatini
Editora de Texto: Ana Lúcia B. de Lucena

COMERCIAL
Diretor Comercial: Roberto Martins Silveira
Gerente Comercial: Flávio Maculan
Gerente de Circulação: Denise Maria Mozol

PRODUÇÃO
Gerente de Produção: João Stungis
Coordenador de impressão: Atílio Roberto Bonon

Preparador de Texto/Coordenador: Eliel Silveira Cunha
Preparadores de Texto: Alzira Moreira Braz, Ana Maria Dilguerian, Levon Yacubian, Luciano Tasca, Maria Teresa Galluzzi, Maria Teresa Martins Lopes, Paulo Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria de Fátima Cardoso, Nair Lucia de Britto
Paste-up: Anastase Potaris, Balduino F. Leite, Edson Donato

Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, nº 2000 - 3º andar
CEP 01452 - São Paulo - SP - Brasil
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta na AM Produções Gráficas Ltda. e impressa na Divisão Gráfica da Editora Abril S.A.

ALAVANCAS, POLIAS E ROLDANAS

A mecânica é a área da física que estuda as forças que interagem entre os corpos. Utilize o computador para analisar alguns de seus princípios, simulando situações da vida cotidiana.

Os grandes monumentos antigos — como as pirâmides do Egito e os templos incas — sempre nos intrigaram, pois os povos que os construíram não possuíam a avançada tecnologia de que dispomos nos dias de hoje.

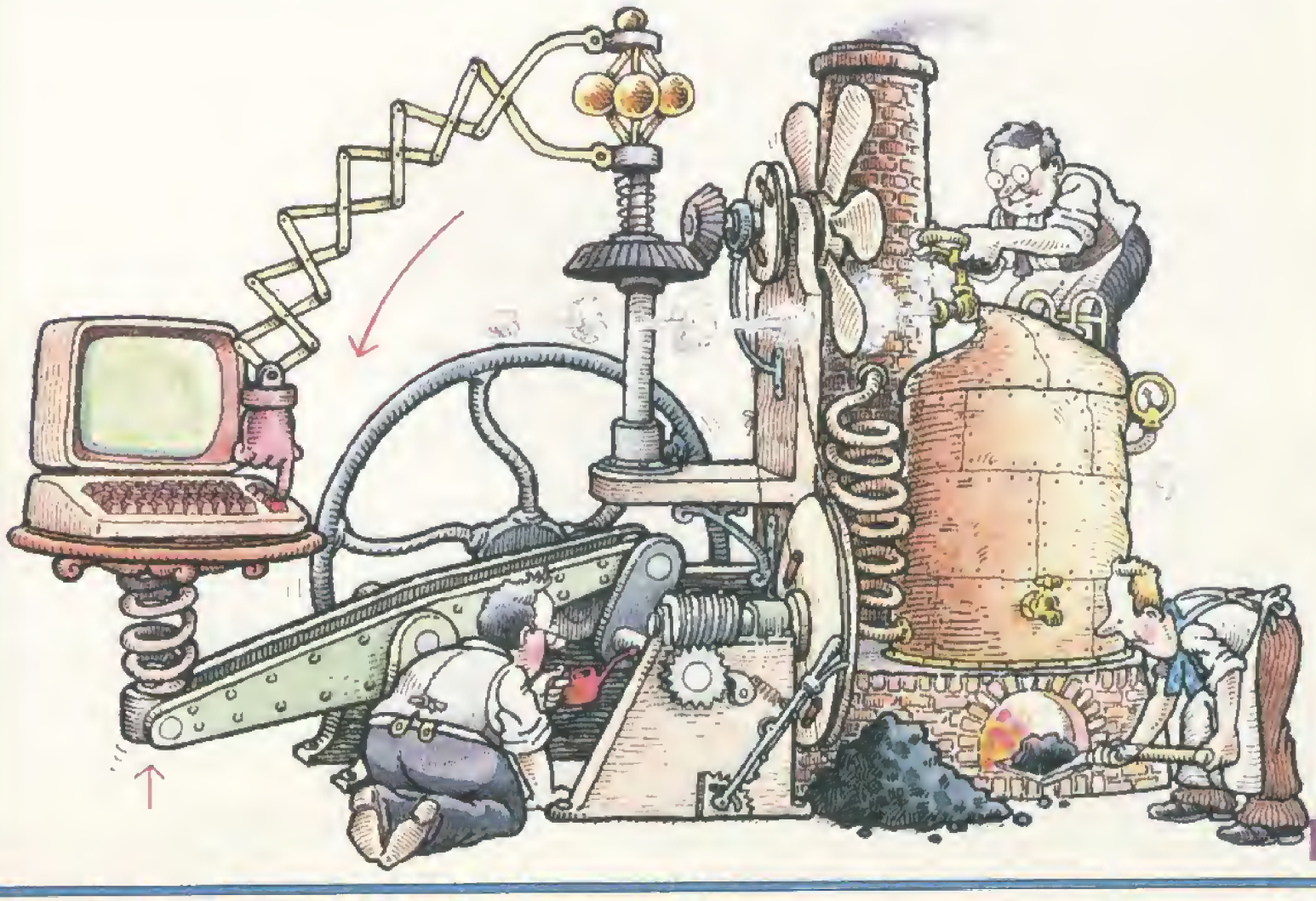
Nenhum construtor contemporâneo se atreveria a executar uma obra como aquelas sem ter em mãos um equipamento que o ajudasse a cortar, transportar e erguer a grandes alturas as gigantescas pedras que as compõem. No entanto, nossas mais sofisticadas máquinas são projetadas a partir dos mesmos princípios que orientavam os antigos engenheiros. A ciência que estuda tais princípios é a mecânica.

As forças que atuam sobre um sistema estático (como um edifício) ou dinâmico (como as engrenagens de uma máquina) são percebidas de modo quase instintivo. Assim, quando levantamos uma xícara de café, calculamos automaticamente a força exata para fazê-lo sem derramar o líquido. Do mesmo modo, ao escolhermos uma tábua para uma estante, sabemos mais ou menos qual a espessura que ela deve ter para suportar um determinado peso.

Em situações simples, como as mencionadas, apenas estimamos as forças em questão. Porém, em alguns casos, os cálculos devem ser muito precisos. É aí que entram os computadores.

Você pode, é claro, usar dados obtidos através de suas próprias experiências para planejar simulações envolvendo forças. Assim, em programas de jogos, é possível estimar, com uma boa margem de acerto, a que distância uma bola vai parar depois de ter sido golpeada por um bastão.

Mas o computador em geral requer dados muito precisos. Quando um engenheiro está projetando uma ponte, por exemplo, ele precisa saber exatamente a força que atua sobre cada estrutura e o seu limite máximo de resis-



tência. Aprendemos, aqui, a calcular forças em sistemas simples, onde as cargas podem ser variáveis.

A ALAVANCA

O transporte de cargas muito pesadas torna-se um sério problema à medida que vão sendo construídas estruturas cada vez maiores. Durante a década de 60, o foguete Saturno V, que levaria o homem até a Lua, era transportado em uma grande plataforma móvel, a maior até então fabricada.

Atualmente, as plataformas de petróleo do mar do Norte são transportadas por terra em veículos que flutuam sobre um colchão de ar. Esses equipamentos, e muitos outros mais simples, dependem sempre de algum tipo de máquina que lhes dê pressão, tração ou movimento. Essas máquinas, por sua vez, não passam de montagens mais complexas de alguns instrumentos básicos, já conhecidos e usados há séculos.

Entre os mais primitivos deles está a alavanca. Em sua forma mais simples, ela consiste de um bastão. Uma das extremidades desse bastão fica embaixo da carga e a outra é levantada para empurrá-la. Usando esse artifício, uma pessoa pode mover objetos muitas vezes mais pesados que ela. Como afirmava Arquimedes, o grande matemático do mundo antigo, poderíamos levantar a Terra, se tivéssemos um bastão bem longo e um ponto de apoio.

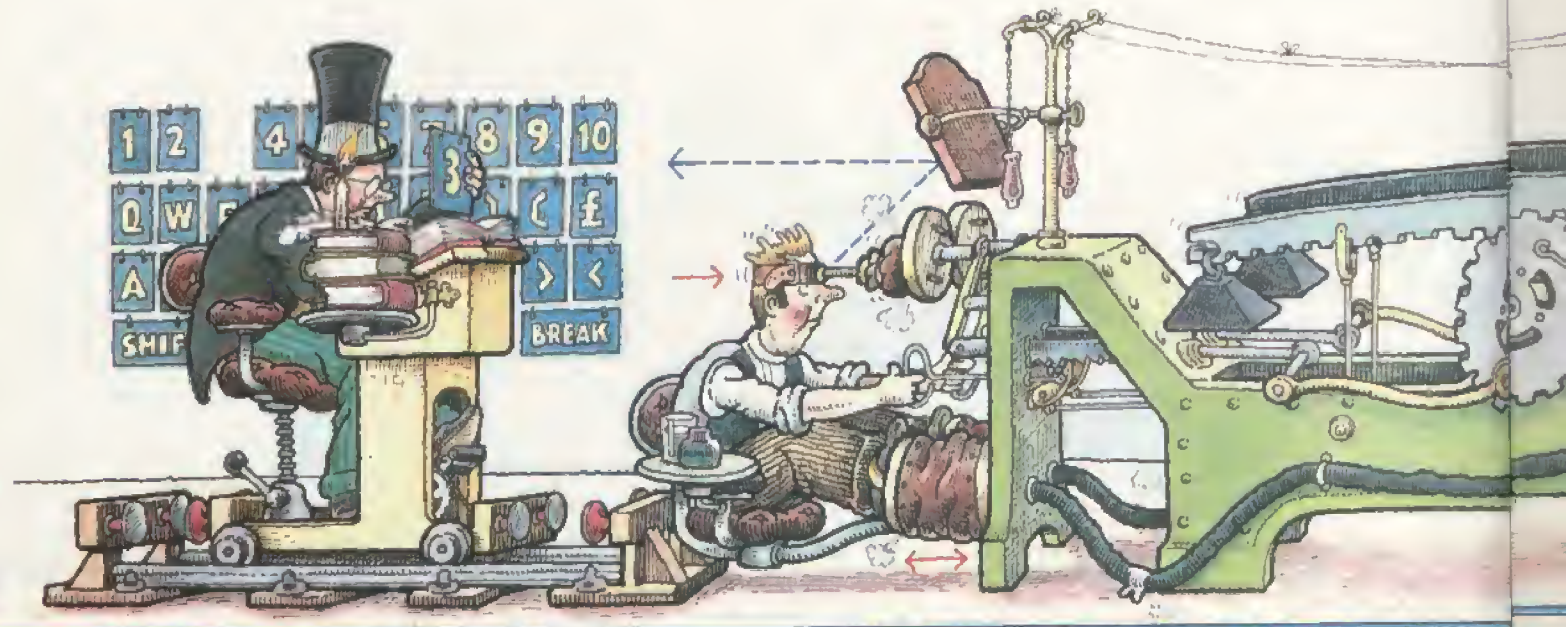
Esse ponto de apoio está situado em algum lugar entre as duas extremidades do bastão. Digite este programa para ver uma simulação do funcionamento de uma alavanca.

S

```
30 BORDER 0: PAPER 0: INK 7:
CLS : OVER 1
40 INPUT "Distancia do apoio
a partir da  esquerda    (1-8
m) ";d
50 IF d<1 OR d>8 THEN  GOTO
40
60 LET w=(10-d)/d
70 PRINT AT 1,0;"Peso necessa
rio para equilibrar 100 KG=";
w*100;"KG"
100 PLOT  INK 4;0,15: DRAW
INK 4;255,0: FOR n=0 TO 55:
PLOT (28+20*d)-n/3,71-n
110 DRAW  INK 7;2*((28+20*d)-
PEEK 23677),0: NEXT n
120 LET a=d-10: LET a=ATN (a/(
1-a*a))+2*ATN (1)
130 FOR b=a TO 2*ATN (1) STEP
(2*ATN (1)-a)/10
140 FOR k=1 TO 2: GOSUB 1000:
GOSUB 1500
160 NEXT k
170 NEXT b
180 LET B=2*ATN (1): GOSUB
1000: GOSUB 1500
190 IF INKEY$="" THEN  GOTO
190
200 RUN
1000 PLOT 120-100*SIN (b),71-20
*d*COS (b)
1011 DRAW 127+100*SIN (b)-PEEK
23677,71+(200-20*d)*COS (b)-PEE
K 23678
1025 DRAW 0,-8: DRAW -10,0: DRA
W 0,-10: DRAW 20,0: DRAW 0,10:
DRAW -9,0: POKE 23678,(PEEK 236
78)+6
1030 RETURN
1500 PLOT 128-100*SIN (b),70-20
*d*COS (b)
1510 LET e=SQR (SQR w)
1520 DRAW 0,-8: DRAW -10*e,0: D
RAW 0,-10*e: DRAW 20*e,0: DRAW
0,10*e: DRAW -9*e,0
1530 RETURN
```

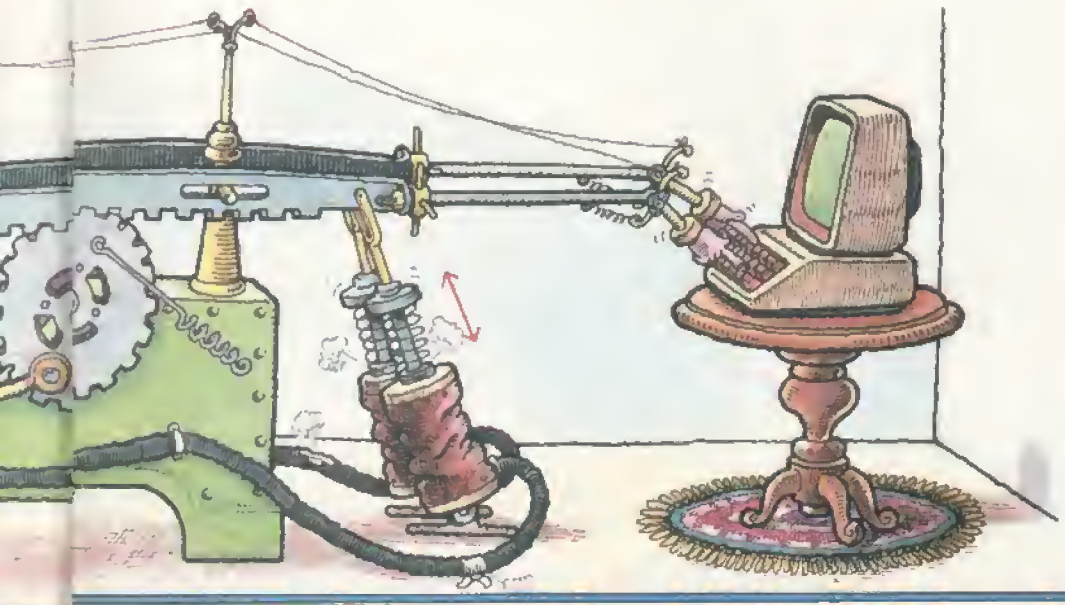
T

```
10 PMODE 3,1:PCLS
20 COLOR 3:LINE (0,180)-(255,19
1),PSET,BF
30 CLS
40 INPUT" DISTANCIA DO APOIO A
PARTIR DA  ESQUERDA (1-8 M)  ";
D
50 IF D<1 OR D>8 THEN 30
60 W=(10-D)/D
70 PRINT:PRINT"PESO NECESSARIO
PARA EQUILIBRAR 100 KG =";W*100
;"KG"
80 FOR G=1 TO 4000:NEXT
90 SCREEN 1,0
100 COLOR 2:LINE(28+20*D,150)-(
18+20*D,179),PSET
110 LINE -(38+20*D,179),PSET:LI
NE -(28+20*D,150),PSET:PAINT(28
+20*D,160),2
120 A=10-D:A=ATN(A*A-1)
130 FOR AN=A TO 2*ATN(1) STEP (
2*ATN(1)-A)/10
140 C=4:GOSUB 1000:C=3:GOSUB 15
00
150 FOR G=1 TO 500:NEXT
160 C=1:GOSUB 1000:GOSUB 1500
170 NEXT
180 AN=2*ATN(1):C=4:GOSUB 1000:
C=3:GOSUB 1500
190 IF INKEY$="" THEN 190
200 RUN
1000 COLOR C:LINE(128-100*SIN(A
N),149-20*D*COS(AN))-(134+100*S
IN(AN),149+(200-20*D)*COS(AN)),
PSET
1020 DRAW"D2L6D6R10U6L8"
1030 RETURN
1500 COLOR C:LINE(132-100*SIN(A
N),147-20*D*COS(AN))-(132-100*S
IN(AN)-SQR(W)*7,147-20*D*COS(AN
)-SQR(W)*7),PSET,BF
1530 RETURN
```



```
5 SCREEN 0:MAXFILES=1:COLOR 15,
1,1
10 INPUT"distância do ponto de
apoio (1-8m) ";D
15 IF D<1 OR D>8 THEN 10
20 W=(10-D)/D
25 SCREEN 2:OPEN"GRP:" FOR OUTP
UT AS #1
30 PRESET(0,0)
35 PRINT #1,"PESO QUE EQUILIBRA
 100 KG=";INT(W*100);"KG"
40 LINE(0,180)-(255,191),3,BF
45 FOR I=0 TO 10
50 LINE (28+20*I,180)-(28+20*I,
191),6
55 NEXT I
60 LINE(28+20*D,150)-(18+20*D,1
79),2:LINE-(38+20*D,179),2:LINE
-(28+20*D,150),2:PAINT(28+20*D,
160),2
65 A=10-D:A=ATN(A*A-1)
70 FOR AN=A TO 2*ATN(1) STEP (2
*ATN(1)-A)/10
80 C=4:GOSUB 1000:C=3:GOSUB 1500
90 FOR G=1 TO 500:NEXT G
100 C=1:GOSUB 1000:GOSUB 1500
110 NEXT AN
120 AN=2*ATN(1):C=4:GOSUB 1000:
C=3:GOSUB 1500
130 IF INKEY$="" THEN 130
140 RUN
1000 LINE (128-100*SIN(AN),149-
20*D*COS(AN))-(134+100*SIN(AN),
149+(200-20*D)*COS(AN)),C
1010 DRAW"D2L6D6R10U6L8"
1020 RETURN
1500 LINE (132-100*SIN(AN),147-
20*D*COS(AN))-(132-100*SIN(AN)-
SQR(W)*7,147-20*D*COS(AN)-SQR(W
)*7),C,BF
1510 RETURN
```

```
5  HOME : HGR : HCOLOR= 3
10  VTAB (23): INPUT "DISTANCIA
 DO PONTO DE APOIO(1-8M)=":D
15  IF D < 1 OR D > 8 THEN 5
20 W = (10 - D) / 10
30  VTAB (24): PRINT "PARA EQUI
LIBRAR 100 KG E PRECISO ";W * 1
00;" KG"
40  HPLOT 28 + 20 * D,120 TO 18
 + 20 * D,149
50  HPLOT  TO 38 + 20 * D,149 T
O 28 + 20 * D,120
60 A = 10 - D:A =  ATN (A * A -
 1)
70  FOR AN = A TO 2 *  ATN (1)
STEP (2 *  ATN (1) - A) / 10
80  HCOLOR= 3: GOSUB 1000
90  FOR G = 1 TO 1000: NEXT G
100  HCOLOR= 0: GOSUB 1000
110  NEXT AN
120 AN = 2 *  ATN (1): HCOLOR=
3: GOSUB 1000
130  GET A$: RUN
1000 X = 128 - 100 *  SIN (AN):
Y = 119 - 20 * D *  COS (AN):R
= 134 + 100 *  SIN (AN):S = 119
 + (200 - 20 * D) *  COS (AN)
1010  HPLOT X,Y TO R,S
1020  HPLOT R,S TO R,S + 2 TO R
 - 6,S + 2 TO R - 6,S + 8 TO R
+ 6,S + 8 TO R + 6,S + 2 TO R,S
 + 2
1500  HPLOT 132 - 100 *  SIN (A
N),117 - 20 * D *  COS (AN) TO
132 - 100 *  SIN (AN),117 - 20
* D *  COS (AN) -  SQR (W) * 7
TO 132 - 100 *  SIN (AN) -  SQR
 (W) * 7,117 - 20 * D *  COS (A
N) -  SQR (W) * 7
1510  HPLOT 132 - 100 *  SIN (A
N) -  SQR (W) * 7,117 - 20 * D
*  COS (AN) -  SQR (W) * 7 TO 1
32 - 100 *  SIN (AN) -  SQR (W)
 * 7,117 - 20 * D *  COS (AN) T
O 132 - 100 *  SIN (AN),117 - 2
0 * D *  COS (AN)
1520  RETURN
```



Ao executar o programa, o computador pedirá que você digite a distância do ponto de apoio **(D)**, a partir da extremidade esquerda de uma barra de 10 m. Ele imprime, então, o peso necessário para equilibrar 100 kg na outra ponta **(W)**. O ponto de apoio é desenhado pelas linhas 100 e 110 (TRS-Color e Spectrum), 40 e 50 (Apple e TK-2000) e 60 (MSX). Um laço controla a animação da alavanca — linhas 130 a 170 (TRS-Color e Spectrum) e 70 a 110 (MSX, Apple e TK-2000) —, chamando a sub-rotina 1000, que desenha a barra, e a 1500, que desenha os pesos.

Introduza diferentes valores para **D** e note que, quanto mais distante do ponto de apoio, menor é o peso necessário. Do mesmo modo, uma distância mais curta requer um peso maior.

DISTÂNCIA X PESO

Existe uma fórmula matemática bem simples para o cálculo de distâncias e pesos em alavancas; basta que você saiba onde se encontra o ponto de apoio. Segundo essa fórmula, a carga, multiplicada pela distância até o ponto de apoio, é igual ao peso usado vezes a distância até esse ponto. Em termos de variáveis, seria o seguinte:

$$C \times DC = P \times DP$$

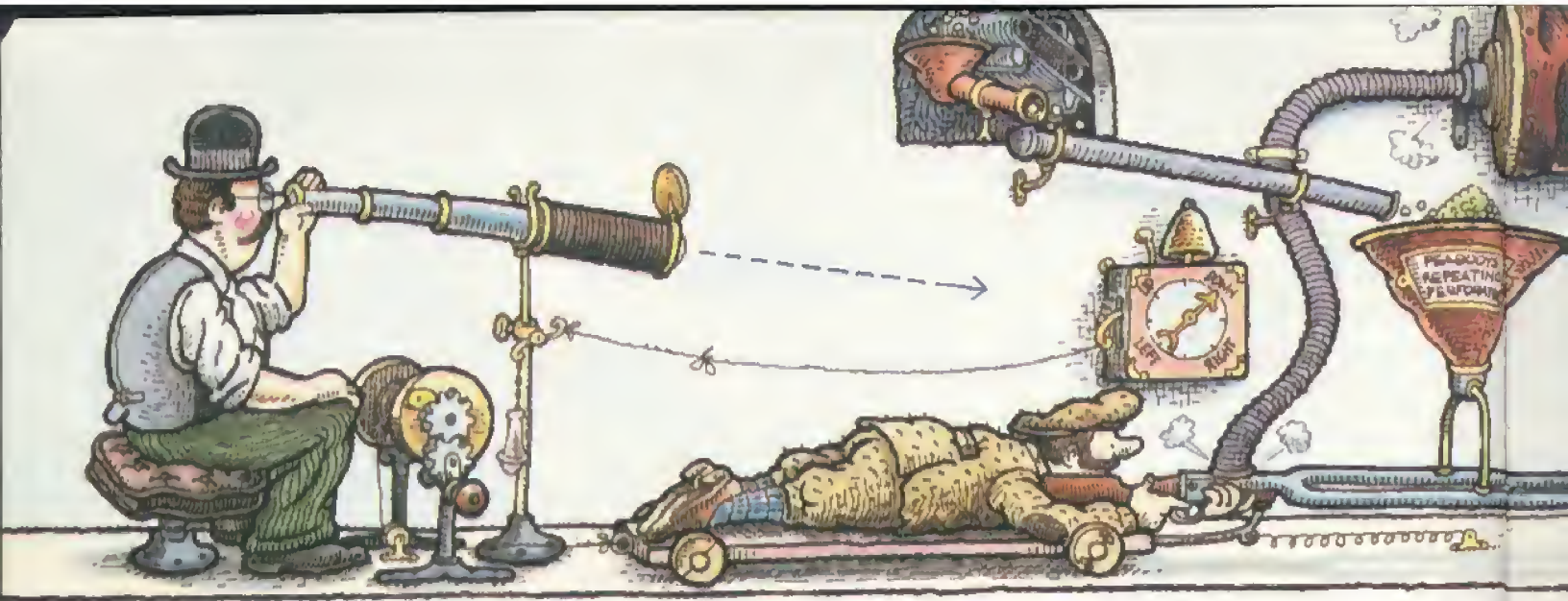
No programa anterior, usamos **W** no lugar de **P**. A barra mede 10 m, logo, **DC** é igual a 10 menos **DP**. Se a carga fosse de 1 kg, o peso necessário para equilibrá-la seria então:

$$W = (10 - DE)/DE$$

conforme usamos nas linhas 60 (TRS-Color e Spectrum) e 20 (MSX, Apple e TK-2000). A variável **DE** (**D**, no programa) foi introduzida por você.

As balanças mecânicas utilizam esse princípio. Toma-se um ponto de apoio no meio da barra e coloca-se o peso a ser medido em uma das extremidades. A partir daí, adicionam-se massas já conhecidas do outro lado, até atingir o equilíbrio. O peso do objeto será a soma dessas massas. Isso funciona muito bem para pequenos pesos, mas, quando se trata de grandes cargas, o procedimento é outro. Nesse caso, o ponto de apoio é colocado bem próximo da carga e calcula-se o peso aplicando a fórmula já explicada.

Depois de ter brincado um pouco com seu programa, você estará pronto para identificar o funcionamento de alavancas em qualquer tipo de máquina.



Ao trabalhar com uma chave inglesa ou com o macaco do seu carro, nunca se esqueça de que a distância que você tomar do ponto de apoio poderá reduzir muito o seu esforço.

SISTEMAS DE POLIAS

As alavancas são muito úteis para puxar e empurrar objetos, mas, quando se quer levar uma carga a grandes alturas, é necessário utilizar as polias. Digite o programa a seguir para vê-las em funcionamento.

```
 10 CLEAR 32399: RESTORE :
 GOSUB 510: BORDER 0: PAPER 0:
  INK 7: CLS
 20 PRINT AT 10,0;"Quantas rol
 danas (2, 4 ou 6)?"
 30 LET a$=INKEY$: IF a$<>"2"
 AND a$<>"4" AND a$<>"6" THEN
 GOTO 30
 40 PRINT TAB (10);a$: LET np=
 VAL (a$): LET l=25*np-50
 45 IF np=2 THEN  POKE 32431,
 25
 46 IF np=4 THEN  POKE 32431,
 17
 47 IF np=6 THEN  POKE 32431,
 14
 50 PRINT : PRINT "Sao necessa
 rios ";INT (1000/np);" quilog
 ramas": PRINT "para levantar
 1 tonelada"
 55 FOR m=1 TO 500: NEXT m
 60 CLS : GOSUB 1000
 70 LET sp=120
 90 FOR k=1 TO 50
100 RAND USR 32400
105 PLOT  OVER 1;191-l-np,sp:
DRAW  OVER 1;8,0
110 PLOT  OVER 1;191-l-np,sp:
DRAW  OVER 1;8,0
120 LET sp=sp-np: IF sp<=0
THEN  LET sp=120
150 NEXT k
160 IF INKEY$="" THEN  GOTO
160
170 RUN
510 FOR n=32400 TO 32491
520 READ a: POKE n,a
530 NEXT n
540 DATA 62,60,79,230,192,15,
15,15,198,64,103,121,230,7,132
,103,121,135,135,230,224,111,
62
550 DATA 175,254,192,208,145,
216,8,14,14,125,177,111,62,30,
254,32,208,145,216,60,79,6,0,
197,229,17,224,91,237,176,225
560 DATA 193,217,8,167,40,30,
71,217,124,60,87,93,230,7,32,
10,123,198,32,95,56,4,122,214,
8,87,235
570 DATA 229,197,237,176,193,
225,217,16,227,217,201
580 RETURN
1000 PLOT 0,170: DRAW 255,0
1010 FOR k=1 TO np STEP 2
1020 CIRCLE (232-k*26),140,13
1030 CIRCLE (258-k*26),50,13
1040 NEXT k
1050 PLOT 245,170: DRAW 0,-125
1060 FOR k=1 TO np-1
1070 PLOT 246-k*26,140: DRAW 0,
-90
1080 NEXT k
1090 PLOT 197-1-np,140: DRAW 0,
-140
1100 PLOT 208,141: DRAW (256-np
*26)-PEEK 23677,0
1110 PLOT 234,50: DRAW (282-np*
26)-PEEK 23677,0
1120 PLOT 206,170: DRAW 0,-28:
PLOT 258-np*26,170: DRAW 0,-28
1130 PLOT 231-1*2/3,50: DRAW 0,
-20
1140 PLOT 232-1/3,50: DRAW 0,-2
0
1145 DRAW (231-1*2/3)-PEEK 2367
7,0
1150 PLOT 231-1/2,29: DRAW 0,-1
0
1160 DRAW -9,0: DRAW 0,-10: DRA
W 19,0: DRAW 0,10: DRAW -9,0
1170 PLOT 180-1,0: DRAW (180-1-
20/SQR (np))-PEEK 23677,0: DRAW
 0,(20/SQR (np))-PEEK 23678: DR
AW (180-1)-PEEK 23677,0: DRAW 0
,0-PEEK 23678
1210 RETURN
```

```
10 PMODE 3,1:DIM P(280)
20 CLS:PRINT "QUANTAS ROLDANAS
(2, 4 OU 6) ? ";
30 A$=INKEY$:IF A$<>"2" AND A$<
>"4" AND A$<>"6" THEN 30
40 PRINT A$:NP=VAL(A$):L=25*NP-
50
50 PRINT:PRINT"SAO NECESSARIOS
";1000/NP:PRINT"QUILOGRAMAS PAR
A LEVANTAR 1 TON.":FOR G=1 TO 4
000:NEXT
60 PCLS:SCREEN 1,0:GOSUB 1000
70 GET(249,172)-(215-L,106),P,G
:SP=38
80 COLOR 4,1
90 FOR K=106 TO 49 STEP -1
100 PUT (249,K+66)-(215-L,K),P,
PSET
110 LINE(191-L-NP,SP)-(199-L-NP
,SP),PRESET
120 SP=SP+NP:IF SP>191 THEN SP=
38
130 LINE(195-L-NP,33)-(195-L-NP
,191),PSET
140 LINE(191-L-NP,SP)-(199-L-NP
,SP),PSET
150 NEXT
160 IF INKEY$="" THEN 160
170 RUN
1000 LINE(0,0)-(255,10),PSET,BF
1010 FOR K=1 TO NP STEP 2
1020 CIRCLE(232-K*26,33),13,2:P
AINT(232-K*26,33),2
1030 CIRCLE (232-K*26,33),14,4,
1,.5,1
1040 CIRCLE (258-K*26,121),13,2
:PAINT (258-K*26,121),2
1050 CIRCLE (258-K*26,121),14,4
,1,0,.5
1060 NEXT
1070 LINE(245,10)-(245,121),PSE
T
1080 FOR K=1 TO NP-1
1090 LINE (246-K*26,33)-(246-K*
26,121),PSET
1100 NEXT
```

```
1110 LINE(195-L-NP,33)-(195-L-N
P,191),PSET
1120 COLOR 4,3:LINE(208,32)-(25
6-NP*26,34),PRESET,BF
1130 LINE(234,120)-(282-NP*26,1
22),PRESET,BF
1140 LINE(206,10)-(206,33),PSET
:LINE(258-NP*26,10)-(258-NP*26,
33),PSET
1150 LINE(232-L*2/3,121)-(232-L
*2/3,141),PRESET
1160 LINE(232-L/3,121)-(232-L/3
,141),PRESET
1170 LINE -(232-L*2/3,141),PRES
ET
1180 LINE(232-L/2,141)-(232-L/2
,151),PRESET
1190 DRAW"L9D19R19U19L9"
1200 LINE(180-L,191)-(180-L-20/
SQR(NP),191-20/SQR(NP)),PRESET,
BF
1210 RETURN
```

```
100  SCREEN 0:COLOR 15,1,1:PRIN
T TAB(7)"Quantas polias(2,4 ou
6)?";
110 A$=INKEY$:IF A$<>"2" AND A$
<>"4" AND A$ <>"6" THEN 110
120 PRINT A$:NP=VAL(A$):L=25*NP
-50
130 PRINT: PRINT TAB(7)"Força n
ecessária =";INT(1000/NP);"Kg":
PRINT TAB(7)"para levantar 1 to
nelada":FOR G=1 TO 4000:NEXT
1000 CLS:SCREEN 2
1010 LINE(0,0)-(255,10),5,BF
1020 FOR K=1 TO NP STEP 2
1030 CIRCLE(232-K*26,33),13,2:P
AINT(232-K*26,33),2
1040 LINE(232-K*26,33)-(232-K*2
6,10)
1050 NEXT K
1060 LINE(208,32)-(256-NP*26,34
),,BF
1070 FOR V=0 TO -30 STEP -5
1080 GOSUB 1500
1090 FOR I=1 TO 300:NEXT I
1100 NEXT V
1110 FOR I=1 TO 2000:NEXT I
1120 GOTO 100
```

```
1500 LINE(194-L-NP-12/NP,75-3*(
V+5))-(196-L-NP+12/NP,75+30/NP-
3*(V+5)),1,BF
1510 LINE(195-L-NP,33)-(195-L-N
P,75-3*V),2
1520 LINE(194-L-NP-12/NP,75-3*V
)-(196-L-NP+12/NP,75+30/NP-3*V)
,10,BF
1530 LINE(249,175+V)-(215-L,107
+V),1,BF
1540 FOR K=1 TO NP STEP 2
1550 CIRCLE(258-K*26,121+V),13,
2:PAINT(258-K*26,121+V),2
1560 NEXT K
1570 LINE(245,10+V)-(245,121+V)
,2
1580 FOR K=1 TO NP-1
1590 LINE(246-K*26,35)-(246-K*2
6,121+V),2
1600 NEXT K
1610 LINE(234,120+V)-(282-NP*26
,122+V),,BF
1620 LINE(232-L*2/3,121+V)-(232
-L*2/3,141+V)
1630 LINE(232-L/3,121+V)-(232-L
/3,141+V)
1640 LINE-(232-L*2/3,141+V)
1650 LINE(232-L/2,141+V)-(232-L
/2,151+V)
1660 DRAW"C8L9D19R19U19L9"
1670 RETURN
```

```
10 E = 780
20 F =  INT (E / 256)
30  POKE 232,E - F * 256: POKE
233,F
40  FOR I = E TO E + 41 + 1 * 3
2
50  READ A: POKE I,A
60  NEXT
70  SCALE= 1: ROT= 0
80  DATA   20 ,0 ,42 ,0 ,74 ,0
,106 ,0 ,138 ,0 ,170 ,0 ,202 ,0
 ,234 ,0 ,10 ,1 ,42 ,1 ,74 ,1 ,
106 ,1 ,138 ,1 ,170 ,1 ,202 ,1
,234 ,1 ,10 ,2 ,42 ,2 ,74 ,2 ,1
06 ,2 ,138 ,2
90  DATA    0 ,72 ,45 ,109 ,209
 ,251 ,219 ,23 ,77 ,73 ,169 ,25
1 ,219 ,27 ,110 ,73 ,9 ,213 ,22
```

```
3 ,219 ,115 ,77 ,9 ,141 ,219 ,6
3 ,255 ,2 ,0 ,0 ,0 ,0
100  TEXT : HOME : VTAB (10): H
TAB (7): PRINT "QUANTAS POLIAS
? (2,4 OU 6) ";: GET NP
110  IF NP <  > 2 AND NP <  > 4
 AND NP <  > 6 THEN 100
120  PRINT NP
130  HGR : HCOLOR= 3: VTAB (23)
: PRINT "FORCA NECESSARIA="; IN
T (1000 / NP);" Kg": PRINT "PAR
A LEVANTAR 1 TONELADA"
1000  FOR I = 0 TO 9
1010  HPLOT 0,I TO 255,I
1020  NEXT I
1030  FOR K = 1 TO NP STEP 2
1040  DRAW 1 AT 147 - K * 8,33
1060  NEXT K
1070  FOR I = 1 TO NP STEP 2
1080  HPLOT 135 - (I - 2) * 8,1
0 TO 135 - (I - 2) * 8,34
1090  NEXT I
1100  HPLOT 143,34 TO 143 - (NP
 - 2) * 8,34
1110  FOR V = 0 TO 36 STEP 3
1115  HCOLOR= 3: GOSUB 1500
1120  FOR T = 1 TO 500: NEXT T
1125  IF V = 36 THEN  GOTO 1150
1130  HCOLOR= 0: GOSUB 1500
1140  NEXT V
1150  FOR T = 1 TO 2000: NEXT T

1160  GOTO 100
1500  FOR K = 1 TO NP STEP 2
1510  DRAW 1 AT 155 - K * 8,121
 - V
1520  NEXT K
1530  HPLOT 155,10 TO 155,123 -
 V
1540  FOR K = 1 TO NP - 1
1550  HPLOT 155 - K * 8,34 TO 1
55 - K * 8,123 - V
1560  NEXT K
1570  HPLOT 138 - (NP - 2) * 8,
34 TO 138 - (NP - 2) * 8,38 + N
P / 2 * V
1580  FOR I =  - 1 TO 1
1590  HPLOT 138 - (NP - 2) * 8
+ I,38 + NP / 2 * V TO 138 - (N
P - 2) * 8 + I,38 + 36 / NP + N
P / 2 * V
1600  NEXT I
1610  HPLOT 151,123 - V TO 151
- (NP - 2) * 8,123 - V
1620  HPLOT 151 - (NP / 2 - 1)
* 8,123 - V TO 151 - (NP / 2 -
1) * 8,128 - V
1630  FOR I =  - 4 TO 4
1640  HPLOT 151 - (NP / 2 - 1)
* 8 + I,128 - V TO 151 - (NP /
2 - 1) * 8 + I,135 - V
1650  NEXT I
1660  RETURN
```

O programa começa perguntando se você quer um sistema de duas, quatro ou seis polias. Em seguida ele mostra na tela o peso necessário para levantar uma carga de 1000 kg. A partir da linha 1000, o computador desenha as roldanas fixas, e, na linha 1500 (na versão para o MSX, Apple e TK-2000) ou 90 (TRS-Color e Spectrum), ele constrói e movimenta as roldanas e os pesos.

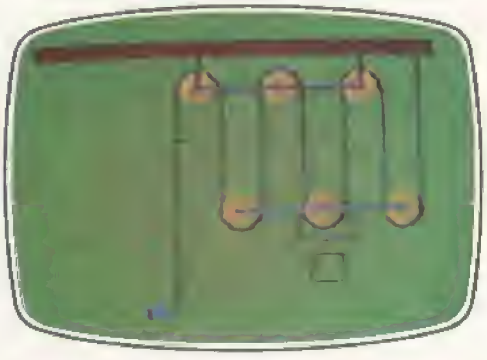
No programa para o Apple e o TK-2000, as linhas 10 a 90 introduzem uma tabela de dados que permite desenhar círculos por meio do comando **DRAW**.

Ao executar o programa, você observará que quanto maior for o número de polias menor será o peso necessário para levantar a carga. Para calcular o peso em um sistema desse tipo, basta dividir a carga pelo número de polias. Se você quiser levantar 1000 kg, usando seis polias, deverá empregar um peso de aproximadamente 166 kg. Em alguns casos ocorre o contrário: sabe-se quanto pesa a carga e qual a força máxima disponível. Nesse caso, divide-se a carga pela força e arredonda-se o resultado para o maior número par. Por exemplo, você sabe que consegue levantar 60 kg usando só uma polia; logo, para levantar 250 kg, você deverá utilizar seis polias.

POUPANDO ESFORÇOS

Como no caso da alavanca, também em um sistema de polias a força está relacionada com a distância. Você pode notar, em nossa simulação, que quanto maior é o número de polias mais a extremidade livre da corda se aproxima do chão. Isso mostra que, para levantar um peso a uma certa altura usando uma força menor, você terá que puxar, para compensar, uma corda de comprimento maior. Na verdade, em um sistema desse tipo, o número de polias já determina qual será o comprimento da corda que precisaremos puxar para que a carga suba em uma unidade. Por exemplo, em um sistema de duas polias, temos que puxar dois metros de corda para que a carga suba um metro.

O sistema que nosso programa simula na tela do microcomputador não é o mais comum. Em geral, as barras passam pelo centro das roldanas, mantendo-as paralelas e bem próximas. Um sistema de seis polias seria visto, então, como três discos colados um ao outro, na vertical, e, abaixo deles, pendurados pelas cordas, mais três discos, onde se prende a carga.



Simulação das polias no MSX.

O ELEVADOR HIDRÁULICO

Outro instrumento que funciona da mesma maneira que a alavanca é o elevador hidráulico. Ele é utilizado para levantar grandes cargas — de automóveis até foguetes e aviões.

O freio dos automóveis atuais, que funciona baseado em um princípio idêntico, mostra como a força de nossos pés pode ser amplificada a ponto de parar, em um tempo muito reduzido, o movimento de um corpo tão pesado.

SIMULAÇÃO

Para observar o funcionamento do elevador hidráulico, digite e rode o programa a seguir.

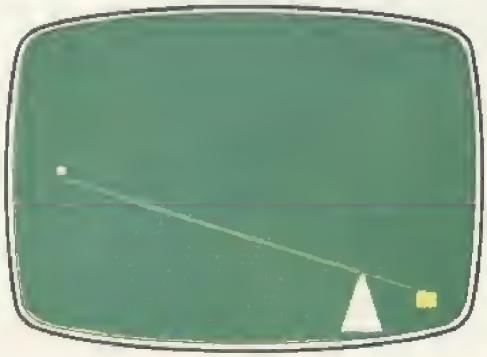
```
 30 BORDER 0: PAPER 7: INK 0:
 CLS
 50 GOSUB 300
 90 INPUT "Curso do embolo (1-
 90) ?  ";tr
100 IF tr<1 OR tr>90 THEN
 GOTO 90
110 FOR k=1 TO tr
120 PLOT 40,128-(k-1): DRAW
INK 7;10,0: PLOT 40,128-k:
DRAW 15,0
130 PLOT 175,127+(k-1)/10:
DRAW  INK 1;56,0: PLOT 175,127
+k/10: DRAW  INK 1;56,0
135 PRINT  INK 0;AT 3,5;k; INK
0;AT 3,25;INT (k/10)
140 NEXT k
150 INK 0: PRINT AT 21,0;"
Novamente  ? (s  OR n)       "
160 IF INKEY$="s" THEN  RUN
170 IF INKEY$<>"n" THEN  GOTO
160
180 STOP
300 FOR n=6 TO 18: PRINT
PAPER 1;AT n,5;"  ": NEXT n
310 FOR n=6 TO 18: PRINT
PAPER 1;AT n,22;"        ":
NEXT n
320 FOR n=18 TO 21: PRINT
PAPER 1;AT n,5;"
        ": NEXT n
330 PLOT 39,155: DRAW 0,-155:
DRAW 192,0: DRAW 0,155: PLOT
56,155: DRAW 0,-124: DRAW 120,
0: DRAW 0,124
340 PLOT 40,127: DRAW -2,0:
PRINT AT 6,3;"0": PLOT 40,37:
DRAW -2,0: PRINT AT 17,2;"90":
 PLOT 232,127: DRAW 2,0: PRINT
```

```
AT 6,30;"0"
350 PLOT 232,137: DRAW 2,0:
PRINT AT 4,30;"9"
360 FOR n=127 TO 37 STEP -10
370 PLOT 40,n: DRAW -2,0: NEXT
n
380 PLOT 232,132: DRAW 2,0
390 PLOT 120,35: DRAW 0,-35
400 PLOT 110,40: DRAW 20,0:
DRAW -5,5: DRAW 5,-5: DRAW -5,
-5
410 INK 7
440 RETURN
```

```
10 PMODE 3,1
20 PCLS
30 SCREEN 1,0
40 DIM P(4),R(31)
50 GOSUB 300
60 GET(32,52)-(43,38),P,G
70 GET(182,52)-(223,23),R,G
80 IF INKEY$="" THEN 80
90 CLS:INPUT"DESLOCAMENTO DO EM
BOLO   (1-90)";TR
100 IF TR<1 OR TR>90 THEN 90
110 SCREEN 1,0:FOR K=1 TO TR
120 PUT(32,52+K)-(43,38+K),P,PS
ET
130 PUT(182,52-K/10)-(223,23-K/
10),R,PSET
140 NEXT
150 IF INKEY$="" THEN 150
160 CLS:PRINT " NOVAMENTE (S/N)
 ?"
170 A$=INKEY$:IF A$<>"S" AND A$
<>"N" THEN 170
180 IF A$="S" THEN RUN
190 CLS:END
300 DRAW"BM30,30C2D131R195U131B
L45D20NR45D96L135U96NL15U20"
310 PAINT(200,70),3,2
320 DRAW"BM32,50C3R10BR140R41"
330 DRAW"BM127,161C2U20BH8C4R16
NH5G5"
340 LINE(34,48)-(41,41),PSET,BF
350 LINE(191,48)-(215,23),PSET,
BF
360 FOR K=0 TO 9
370 COLOR 2:LINE(26,50+K*10)-(3
0,50+K*10),PSET
380 NEXT
390 DRAW"BM20,46C4D6L4U6R4BD90N
L4D6L4U6BL4ND6L4D3R4C2"
400 FOR K=0 TO 9 STEP 3
410 LINE(225,50-K)-(229,50-K),P
SET
420 NEXT
430 DRAW"BM232,48C4D6R4U6L4BU9N
R4U3R4D6"
440 RETURN
```

```
10 SCREEN 0:COLOR 1,14,14
20 LOCATE 3,10:INPUT"Qual o cur
so do êmbolo(1-90)";C
30 IF C<1 OR C>90 THEN 20
40 GOSUB 300
50 FOR I=0 TO C-1
60 LINE(31,37+I)-(44,50+I),14,B
F
70 LINE(34,50+I)-(39,43+I),15,B
F
80 LINE(181,50-I/10)-(225,50-I/
10),2
90 LINE(191,23-I/10)-(215,48-I/
10),15,BF
100 NEXT I
110 IF INKEY$="" THEN 110 ELSE
GOTO 10
300 SCREEN 2:COLOR 15,14,14
310 DRAW"BM30,30C2D131R195U131B
L45D20NR45D96L135U96NL15U20"
320 PAINT(200,70),2,2
330 DRAW"BM32,50C3R10BR140R41"
340 DRAW"BM127,161C2U20BH8C4R16
NH5G5"
350 LINE(34,48)-(39,41),,BF
360 LINE(191,48)-(215,23),,BF
370 FOR K=0 TO 9
380 LINE(26,50+K*10)-(30,50+K*1
0),2
390 NEXT K
400 DRAW"BM20,46C4D6L4U6R4BD90N
L4D6L4U6BL4ND6L4D3R4C2"
410 FOR K=0 TO 9 STEP 3
420 LINE(225,50-K)-(229,50-K)
430 NEXT K
440 DRAW"BM232,48C4D6R4U6L4BU9N
R4U3R4D6"
450 RETURN
```

```
20  HOME : HTAB (5): VTAB (10)
30  INPUT "QUAL O CURSO DO EMBO
LO(1-90)? ";C
40  GOSUB 300
50  FOR I = 1 TO C - 1
55  HCOLOR= 0
60  HPLOT 31,49 + I TO 44,49 +
I
70  HPLOT 34,42 + I TO 39,42 +
I
80  HCOLOR= 3
90  HPLOT 34,49 + I TO 39,49 +
I
100  HPLOT 181,50 - I / 10 TO 2
24,50 - I / 10
110  HPLOT 191,25 - I / 10 TO 2
15,25 - I / 10
120  NEXT I
130  FOR T = 1 TO 5000: NEXT T:
 TEXT : GOTO 20
300  HGR2 : HCOLOR= 3
310  HPLOT 30,30 TO 30,161 TO 2
25,161 TO 225,30
320  HPLOT 180,30 TO 180,146 TO
 45,146 TO 45,30
330  FOR I = 0 TO 96
340  HPLOT 30,50 + I TO 45,50 +
 I
350  HPLOT 180,50 + I TO 225,50
 + I
360  NEXT I
370  FOR I = 0 TO 15
380  HPLOT 30,146 + I TO 225,14
6 + I
390  NEXT I
400  FOR I = 0 TO 9
410  HPLOT 26,50 + I * 10 TO 30
,50 + I * 10
420  NEXT I
430  FOR I = 0 TO 9 STEP 3
440  HPLOT 225,50 - I TO 229,50
 - I
450  NEXT I
460  HPLOT 20,46 TO 20,53 TO 15
,53 TO 15,46 TO 19,46
470  HPLOT 11,140 TO 7,140 TO 7
,136 TO 11,136 TO 11,143 TO 7,1
43
480  HPLOT 20,136 TO 20,143 TO
15,143 TO 15,136 TO 20,136
490  HPLOT 232,48 TO 237,48 TO
237,55 TO 232,55 TO 232,48
500  HPLOT 232,42 TO 237,42 TO
237,35 TO 232,35 TO 232,39 TO 2
37,39
510  HPLOT 103,140 TO 117,140 T
O 110,136: HPLOT 117,140 TO 110
,144
530  FOR J = 1 TO 14
540  HPLOT 34,43 + J TO 39,43 +
 J
550  NEXT J
560  FOR K = 1 TO 26
570  HPLOT 191,24 + K TO 215,24
 + K
580  NEXT K
590  RETURN
```



A alavanca na tela do TRS-Color.

Inicialmente, o computador pergunta quanto você quer que o êmbolo desça, em um intervalo de 1 a 90. A rotina que começa na linha 300 desenhará o elevador e o laço da linha 110 (na versão para os micros Spectrum e TRS-Color) ou da 50 (MSX, Apple e TK-2000) deverá movimentá-lo.

PRINCÍPIOS

Quanto maior for o diâmetro do êmbolo maior terá que ser a força aplicada. Na nossa simulação, ele é bem menor que o diâmetro do elevador; portanto, a força para empurrá-lo para baixo será bem menor que o peso do objeto a ser levantado. Observamos, porém, que, assim como no caso das polias, o trajeto do êmbolo é maior que a altura da carga — ou seja, teremos que mover o êmbolo em várias unidades para que o elevador suba uma só.

CONTROLE POR TECLAS MÚLTIPLAS

No controle de jogos, o joystick ganha em sofisticação, enquanto o teclado se destaca pela versatilidade. Veja como fazer para que sejam detectadas várias teclas ao mesmo tempo.

Uma das características mais importantes num jogo é a qualidade de interação computador-usuário. Na maioria dos casos, o joystick oferece um controle mais sofisticado, mas o teclado o supera, sem dúvida alguma, no que diz respeito à versatilidade.

Em geral, utilizam-se os comandos **INKEY$** e **GET$** para detectar se uma tecla está sendo pressionada. Esses comandos, porém, apresentam uma deficiência: só são capazes de detectar duas teclas ao mesmo tempo caso uma delas seja **SHIFT** (ou similares, como **CTRL** ou **SYMBOL/SHIFT**).

A detecção de tais combinações é suficiente para a maior parte dos propósitos, mas não para todos. Se quiséssemos, por exemplo, controlar os movimentos vertical e horizontal de uma nave, disparar o laser e lançar bombas, simultaneamente, seríamos obrigados a acionar quatro teclas. A solução para esse tipo de problema depende do próprio computador.

EFEITO DO ACIONAMENTO DAS TECLAS

Para entender como se dá a detecção múltipla, precisamos examinar os mecanismos que permitem ao computador saber qual tecla está sendo pressionada. Os micros pessoais utilizam, comumente, dois processos. O primeiro consiste em "varrer" o teclado de tempos em tempos, verificando se alguma tecla foi pressionada. No TRS-Color, por exemplo, o teclado é "varrido" a cada 1/100 de segundo; já no MSX, isso ocorre a cada 1/60 de segundo.

No segundo processo, o acionamento de uma tecla gera uma mensagem — chamada *interrupção* —, que é enviada ao processador. O computador interrompe sua tarefa e verifica o teclado, em busca da tecla pressionada. Esse método é bem mais eficiente e versátil que o da varredura, porque dispensa a verificação das teclas quando não estão sendo acionadas — ou seja, o primeiro método varre o teclado de tempos em tempos, mesmo que nenhuma tecla seja acionada; o segundo só faz a varredura quando há acionamento de teclas.

Seja qual for o método utilizado, o computador deve ser capaz de identificar a tecla acionada o mais rapidamente possível. A maioria dos teclados emprega um sistema denominado *gerador de matriz*, fornecendo ao computador um número que identifica a posição da tecla na matriz. O que vai ocorrer depois da geração do número varia de máquina para máquina.

S

O teclado do Spectrum compõe-se de quatro fileiras com dez teclas cada. Supondo que cada fileira seja dividida ao meio, teremos oito fileiras com grupos de cinco teclas. Cada grupo, por sua vez, tem comunicação com uma porta de entrada/saída. O teclado do Spectrum facilita a detecção de várias teclas acionadas ao mesmo tempo, pois possui 65536 dessas portas, identificadas por números de 0 a 65535. A tabela a seguir fornece o endereço da porta para cada um dos oito grupos:

TECLAS	GRUPO	PORTA
V C X Z CAPS/SHIFT	0	65276
G F D S A	1	65022
Q W E R T	2	64510
5 4 3 2 1	3	63486
6 7 8 9 0	4	61438
Y U I O P	5	57342
H J K L ENTER	6	49150
B N M SYM/SHIFT ESPAÇO	7	32766

Para calcular o endereço da porta de cada grupo, use esta fórmula:

254 + 256*(255 − 2↑n)

Nessa fórmula, **n** significa o número do grupo listado na tabela.

Digite e rode o programa a seguir. Ele calcula o endereço da porta para um certo grupo de teclas.

```
10 INPUT "DIGITE NUMERO DO GR
UPO DA TECLA ";n
20 PRINT '"NUMERO DO GRUPO DA
 TECLA ";n
30 PRINT "ENDERECO DA PORTA "
;254+256*(255-2^n)
40 GOTO 10
```

Cada endereço de porta na tabela dada anteriormente assume um valor, dependendo da tecla pressionada.

Para usar corretamente o próximo programa, remova a impressora ou qualquer outra interface. Caso contrário, os resultados serão alterados. Digite estas linhas, rode, e acione de diferentes maneiras as teclas de 1 a 5, inclusive várias ao mesmo tempo:

```
10 PRINT AT 0,0;IN 63486
20 GOTO 10
```

Observe que os números exibidos na tela mudam conforme a combinação de

- O EFEITO DO ACIONAMENTO DAS TECLAS
- COMO FUNCIONA O SEU TECLADO
- OS DIFERENTES MÉTODOS
- O CONTROLE DO JOGO
- A MATRIZ DO TECLADO
- DETECÇÃO MÚLTIPLA DE VÁRIAS TECLAS
- UM TECLADO A QUATRO MÃOS

teclas acionadas. O valor no endereço da porta é armazenado em um único byte, no qual o quinto bit é sempre 1. Normalmente, o sexto e o sétimo bits são 0. Porém, quando houver um sinal no soquete EAR, ou quando o computador esquentar, o sexto bit passa a ser 1. Os bits de 0 a 4 representam as teclas do grupo. Cada bit assume valor 1 quando a tecla correspondente a ele não estiver sendo pressionada e valor 0, quando esta for pressionada. O quarto bit corresponde à tecla da extrema esquerda do grupo e o bit zero à tecla da extrema direita.

Quando um determinado bit contém o valor 1, ele contribui para o valor que está na porta; quando seu valor é 0, não contribui com nada. A contribuição de cada bit para o valor na porta está na tabela a seguir:

BIT	8	7	6	5	4	3	2	1
CONTRIBUIÇÃO	128	64	32	16	8	4	2	1

Portanto, se o sexto bit for 0 (nada conectado em EAR) e a tecla 5 estiver sendo pressionada, a disposição dos bits na porta 63486 será 00101111, o que equivale a 0+0+32+0+8+4+2+1=47. Assim, podemos detectar o acionamento das cinco teclas de um grupo. O mesmo se aplica aos outros grupos de teclas. Poderíamos, por exemplo, acrescentar ao programa linhas que imprimissem na tela o valor de cada porta.

T

As teclas do TRS-Color estão dispostas numa matriz de sete linhas por oito colunas. São necessários nove bytes de memória para armazenar o estado de uma linha e de seus oito elementos. Dos nove bytes, o primeiro contém o estado da linha e os outros oito, o de cada um dos oito elementos dela. O arranjo da matriz e o método usado na decodificação da linha fazem com que, normalmente, seja impossível detectar mais de uma tecla ao mesmo tempo.

No entanto, podemos "enganar" o varredor de teclado, levando-o a não detectar nenhuma tecla numa certa linha, o que, na varredura seguinte, o obrigaria a assumir qualquer tecla como uma nova ocorrência. Para isso, é necessário colocar no primeiro byte o valor hexadecimal FF, produzindo uma alteração no estado das linhas.

Esse método não é muito satisfatório, pois requer freqüentes mudanças ao longo do programa. Mas há uma solução alternativa: chamar uma rotina em linguagem de máquina para forçar uma varredura completa do teclado, sempre que precisarmos. Como você verá, este foi o sistema usado nos programas que apresentaremos mais adiante.

TK

No TK-2000, as teclas estão dispostas num gerador de matriz de oito linhas por seis colunas. Existem dois endereços de memória que trabalham com a matriz: o endereço −16384, ao qual demos o nome de KBOUT, e o −16368, que chamamos de KBIN. KBOUT controla o estado das linhas da matriz. Cada um de seus bits é responsável por uma linha na matriz de tecla. Se um bit de KBOUT possui valor 1, todos os elementos (teclas) da linha correspondente também serão iguais a 1.

KBIN, por sua vez, controla o esta-

O que é *roll-over*?

Esta é uma expressão inglesa que designa uma característica do processador de teclados de algumas linhas de microcomputadores: a possibilidade de armazenar em uma memória intermediária (*buffer* de teclado) as *n* últimas teclas pressionadas.

O número de teclas digitadas que o computador consegue armazenar depende do tamanho do buffer de teclado (se for fixado em hardware) e do software de processamento do teclado. Por isso, falamos em *n-key-roll-over*, ou seja, quantas teclas o roll-over é capaz de processar. Alguns computadores têm roll-over de duas a cinco teclas; outros, de até 128.

Qual é a vantagem de dispor de roll-over no teclado de um micro? Bem, certos softwares (sobretudo processadores de texto) não conseguem acompanhar um ritmo mais acelerado de digitação. Se não houver a capacidade de roll-over, o computador acaba limitando o usuário, que começa a "perder" várias teclas pressionadas rapidamente, levando a erros irritantes de digitação. O roll-over afasta esse risco, pois, mais cedo ou mais tarde, o programa acaba "alcançando" o digitador.

do dos elementos de cada linha. Somente os bits de 0 a 5 de KBIN são usados, já que existem apenas seis elementos em cada linha. Os bits de KBIN só lêem o valor contido nas teclas correspondentes quando estas são pressionadas — ou seja, se nenhuma tecla estiver sendo pressionada, KBIN vale 0 (todos os bits são iguais a zero). Esse endereço requer uma atenção especial, uma vez que lê exclusivamente colunas. Se quisermos saber a que linha pertencem os elementos (teclas) lidos por KBIN, precisaremos identificar o valor contido em KBOUT.

Para forçar a varredura do teclado, foi necessário montar uma rotina em linguagem de máquina. Essa rotina, armazenada a partir do endereço 770 no primeiro programa, procura pelo acionamento de cinco teclas específicas: as quatro setas e a barra de espaço. A pressão sobre qualquer outra tecla não terá nenhum efeito.

A rotina usa os endereços 900 a 904 como indicadores do estado das teclas. Se o valor for 1, a tecla foi acionada; se for 0, não houve pressão sobre ela. Como se pode observar nas linhas 135 a 155, esses endereços são constantemente zerados. Isso é feito para a rotina não "pensar" que a tecla correspondente está sempre pressionada.

Remova o **POKE 900,0** da linha 135 e veja o que ocorre com a mira quando se aperta a tecla ↑. Ela começa a subir e não pára mais. Foi preciso usar endereços (900 a 904) como indicadores porque, coincidentemente, todas as cinco teclas em questão são lidas pelo mesmo bit de KBIN — ou seja, na matriz, elas estão em linhas diferentes mas em colunas iguais.

A rotina do segundo programa funciona de maneira semelhante, detectando seis teclas em vez de cinco. O endereço inicial foi deslocado para 36900 e os indicadores aparecem a partir de 36800. Essa rotina apresenta um aperfeiçoamento: os indicadores são zerados dentro dela própria.

MSX

A matriz que contém as teclas do MSX é maior que a dos outros micros: compõe-se de dez linhas com oito colunas cada, totalizando, assim, oitenta teclas. Nela existem duas portas de controle. A de endereço 170 ativa a linha que desejamos ler, enquanto a de endereço 169 contém o estado das teclas da linha ativada, sendo que cada tecla corresponde a um de seus bits.

Para simplificar, chamaremos a porta que ativa as linhas (170) de porta C, e a que lê as colunas, de porta B. A porta C usa somente os bits de 0 a 3 para ativar seqüencialmente cada uma das dez linhas. Isso é possível graças a um decodificador localizado entre a porta C e a matriz. As linhas da matriz de teclas são numeradas de 0 a 9. Para ativá-las, basta fornecer à porta C o número da linha desejada. Ao se ativar uma linha, todas as suas colunas (teclas) assumem o valor 1. Quando se pressiona uma tecla pertencente a uma linha ativada, o bit da porta B correspondente a essa tecla é zerado. Portanto, se ativarmos uma linha através da porta C, e, logo em seguida, lermos a porta B sem que nenhuma tecla seja acionada, veremos que a porta B tem valor 255, ou seja, todos os oito bits têm valor igual a 1.

O programa a seguir ativa a linha 8 da matriz — que contém as teclas das setas e barra de espaço — e, em seguida, lê a porta B, verificando quais bits foram zerados, isto é, quais teclas foram acionadas.

STM

Os usuários do Apple devem ter notado a ausência de programas para esse micro. É que o método utilizado pelo Apple para detectar teclas torna quase impossível — dentro dos propósitos deste artigo — criar rotinas que detectem acionamento múltiplo de teclas.

Os programas apresentados a seguir permitem-nos mover uma mira pela tela e também atirar. Nos micros TRS-Color, MSX e TK-2000, use as setas para mover e a barra de espaço para atirar. No Spectrum, use as setas para mover e o número 9 para atirar. Os programas do TRS-Color e do TK-2000 requerem um pequeno programa em linguagem de máquina para detectar as teclas.

T

```
10 CLEAR 200,32746
20 FOR K=32747 TO 32767:READ A:
POKE K,A:NEXT
30 DATA 48,140,9,191,1,155,134,
126,183,1,154,57,52,3,134,127,1
83,1,81,53,131
35 EXEC 32747
40 V=247:P=1295:L=1295:CLS3
50 IF PEEK(341)=V AND P>1055 TH
EN P=P-32
60 IF PEEK(342)=V AND P<1504 TH
EN P=P+32
70 IF PEEK(343)=V AND P>1024 TH
EN P=P-1
80 IF PEEK(344)=V AND P<1535 TH
EN P=P+1
90 IF PEEK(345)=V THEN SOUND 20
0,1
100 POKE L,175:POKE P,43:L=P:GO
TO 50
```

Digite e rode o programa do TRS-Color para se certificar de que não há erros. Em seguida, acrescente a linha 35 e rode-o novamente.

```
35 EXEC 32747
```

S

```
 10 BORDER 0: PAPER 0: INK 7
 20 CLS
 30 LET y=11: LET x=15
 40 LET p=63486
 50 GOSUB 220
 60 IF i=31 THEN  GOSUB 290
 70 LET p=61438
 80 GOSUB 220
 90 IF i=59 THEN  GOSUB 380
100 IF i=47 THEN  GOSUB 320
110 IF i=55 THEN  GOSUB 350
120 IF i=61 THEN  GOSUB 410
130 IF i=57 THEN  GOSUB 380:
GOSUB 410
140 IF i=45 THEN  GOSUB 320:
GOSUB 410
150 IF i=53 THEN  GOSUB 350:
GOSUB 410
160 IF i=43 THEN  GOSUB 380:
```

```
GOSUB 320
170 IF i=51 THEN  GOSUB 380:
GOSUB 350
180 IF i=41 THEN  GOSUB 380:
GOSUB 320: GOSUB 410
190 IF i=49 THEN  GOSUB 380:
GOSUB 350: GOSUB 410
200 GOSUB 250
210 GOTO 40
220 LET i=IN p
230 IF i>191 THEN  LET i=i-64
240 RETURN
250 PRINT AT y,x;" + "
260 PRINT AT y+1,x;"   "
270 PRINT AT y-1,x;"   "
280 RETURN
290 IF x<1 THEN  RETURN
300 LET x=x-1
310 RETURN
320 IF y>19 THEN  RETURN
330 LET y=y+1
340 RETURN
350 IF y<2 THEN  RETURN
360 LET y=y-1
370 RETURN
380 IF x>28 THEN  RETURN
390 LET x=x+1
400 RETURN
410 SOUND .004,20
420 SOUND .004,10
430 PRINT AT y,x;"   "
440 RETURN
```

```
10 COLOR 1,5,5:SCREEN2,1
20 X=120:Y=90:H=240:B=169:C=170
30 FOR K=1 TO 8:READ A
40 B$=B$+CHR$(A)
50 NEXT
60 DATA 129,66,60,36,36,60,66,1
29
70 SPRITE$(0)=B$
80 OUT C,(INP(C) AND H)OR 8
90 I=INP(B)
100 IF(I AND 128)=0 THEN X=X+3
110 IF(I AND 64)=0 THEN Y=Y+3
120 IF(I AND 32)=0 THEN Y=Y-3
130 IF(I AND 16)=0 THEN X=X-3
140 IF(I AND 1)=0 THEN PLAY"L64
BAC"
150 PUTSPRITE 0,(X,Y),1,0:GOTO
80
```

```
10  HOME
15  FOR K = 800 TO 804
20  READ A
25  POKE K,A
30  NEXT
35  DATA  64,32,16,8,4
40  FOR K = 770 TO 793
45  READ A
50  POKE K,A
55  NEXT
60  DATA  162,0,189,32,3,141
65  DATA  0,192,169,1,45,16
70  DATA  192,240,3,157,132,3
75  DATA  232,224,5,208,235,96
80  HGR2
85 H = 138:V = 96
90 X = 138:Y = 96
95  HCOLOR= 0
100  HPLOT H,V TO H + 4,V
105  HPLOT H + 2,V - 2 TO H + 2
 TO V + 2
110 H = X:V = Y
115  HCOLOR= 3
120  HPLOT X,Y TO X + 4,Y
125  HPLOT X + 2,Y - 2 TO X + 2
,Y + 2
130  CALL 770
135  IF  PEEK (900) = 1 AND Y >
 3 THEN Y = Y - 2: POKE 900,0
140  IF  PEEK (901) = 1 AND Y <
 188 THEN Y = Y + 2: POKE 901,0

145  IF  PEEK (902) = 1 AND X <
 274 THEN X = X + 2: POKE 902,0

150  IF  PEEK (903) = 1 AND X >
 1 THEN X = X - 2: POKE 903,0
155  IF  PEEK (904) = 1 THEN  P
RINT  CHR$ (7): POKE 904,0
160  GOTO 95
```

UM TECLADO A QUATRO MÃOS

Rodando o programa a seguir, você verá, logo após a introdução, dois seres vestindo "traje lunar", em posição de duelo. Este jogo foi feito para dois jogadores. No Spectrum, as teclas de comando são as seguintes: 1 move o ser da esquerda para cima, Q, para baixo, e A dispara o laser; 0, P e Ê controlam os movimentos do ser da direita. No TRS-Color, o jogador da esquerda usa as setas para cima e para baixo e a tecla Z; o da direita usa −, @ e /. No MSX, o da esquerda usa E, D e F; o da direita usa I, J e H. No TK-2000, finalmente, o jogador da esquerda usa 1, A e Q; o da direita, 0, (:) e P.

```
10 CLS:PMODE 4,1:SS=PEEK(186)*2
56+PEEK(187)
20 FOR K=SS TO SS+480 STEP 32
30 FOR J=K TO K+3:READ A:POKE J
,A:NEXT J,K
40 DATA 1,192,3,128,3,192,3,192
,6,0,0,96,15,192,3,240
50 DATA 31,192,3,248,63,192,3,2
52,15,0,0,240,15,135,113,240
60 DATA 15,248,15,240,15,248,15
,240,15,128,1,240,15,128,1,240
70 DATA 15,128,1,240,12,192,3,4
8,12,192,3,48,14,224,7,112
80 DIM L(6),R(6),B(6)
90 GET(0,0)-(15,15),L:GET(16,0)
-(31,15),R,G
100 PCLS:PRINT @10,"D U E L O"
110 PRINT @98,"GANHE PONTOS ACE
RTANDO O SEU      OPONENTE. CADA
 JOGADOR TEM              SEIS BA
LAS."
120 PRINT:PRINTTAB(6);"C O N T
R O L E S"
130 PRINT:PRINT" JOGADOR 1";TAB
(19);"JOGADOR 2"
140 PRINT:PRINT"   UP      --CIM
A--      -"
150 PRINT"  DOWN     --BAIXO-
  P"
160 PRINT"     Z      --FOGO--
  /"
170 PRINT @482,"qualquer tecla
para comecar";
180 IF INKEY$="" THEN 180
190 X1=16:X2=232:Y1=88:Y2=88:B1
=6:B2=6:S1=0:S2=0:PCLS
200 PUT(X1,Y1)-(X1+15,Y1+15),L,
PSET:PUT(X2,Y2)-(X2+15,Y2+15),R
,PSET
210 FOR K=1 TO 6:CIRCLE(10*K,2)
,1,5:CIRCLE(255-10*K,2),1,5:NEX
T
220 SCREEN 1,1:A$=INKEY$
230 L1=Y1:L2=Y2
240 IF PEEK(341)=247 AND Y1>16
THEN Y1=Y1-8
250 IF PEEK(342)=247 AND Y1<176
 THEN Y1=Y1+8
260 IF PEEK(343)=223 AND Y2>16
THEN Y2=Y2-8
270 IF PEEK(338)=251 AND Y2<176
 THEN Y2=Y2+8
280 IF B1=0 AND B2=0 THEN 360
290 IF L1=Y1 THEN 310
300 PUT (X1,L1)-(X1+15,L1+15),B
:PUT(X1,Y1)-(X1+15,Y1+15),L
310 IF L2=Y2 THEN 330
320 PUT(X2,L2)-(X2+15,L2+15),B:
PUT(X2,Y2)-(X2+15,Y2+15),R
330 IF PEEK(340)=247 GOSUB 1000
340 IF PEEK(345)=223 GOSUB 1500
350 GOTO 230
360 CLS:IF S1>S2 THEN PRINT @96
,"JOGADOR 1 GANHOU POR";S1;"A";
S2:GOTO 390
370 IF S2>S1 THEN PRINT @96,"JO
GADOR 2 GANHOU POR";S2;"A";S1:G
OTO 390
380 PRINT @96,"HOUVE EMPATE :";
S1;"A";S1;"!"
390 A$=INKEY$:GOTO 180
1000 IF B1=0 THEN RETURN
1010 PLAY"T12005AGFEDC"
1020 FOR N=32 TO 232 STEP 16
1030 LINE(N+1,Y1+7)-(N+6,Y1+7),
PSET
1040 LINE(N+1,Y1+7)-(N+6,Y1+7),
PRESET
1050 NEXT
1060 IF Y1=Y2 OR Y1+8=Y2 THEN S
1=S1+1:CIRCLE(10*S1,8),2,5:PLAY
"T801GDBC"
1070 CIRCLE(10*B1,2),1,0:B1=B1-
1
1080 RETURN
1500 IF B2=0 THEN RETURN
1510 PLAY"T12005BAGFEDC"
1520 FOR N=216 TO 32 STEP -16
1530 LINE(N+1,Y2+7)-(N+6,Y2+7),
PSET
1540 LINE(N+1,Y2+7)-(N+6,Y2+7),
PRESET
1550 NEXT
1560 IF Y2=Y1 OR Y2+8=Y1 THEN S
2=S2+1:CIRCLE(255-10*S2,8),2,5:
PLAY"T801GDBC"
```

```
1570 CIRCLE(255-10*B2,2),1,0:B2
=B2-1
1580 RETURN
```

```
 10 BORDER 0: PAPER 0: INK 7
 20 BRIGHT 0: OVER 0: CLS
 30 PRINT AT 1,9; INK 6; FLASH
  1;" D U E L O "
 40 PRINT : PRINT
 50 PRINT  INK 5;"  GANHE PONT
 OS ACERTANDO SEU         OPONE
 NTE. CADA JOGADOR
   TEM SEIS BALAS."
 60 PRINT : PRINT
 70 PRINT TAB 6; INVERSE 1;"C
 O N T R O L E S"
 80 PRINT : PRINT " JOGADOR 1
          JOGADOR 2 "
 90 PRINT : PRINT "     1      -
 -CIMA--      0     "
100 PRINT : PRINT "     Q      -
-BAIXO-      P     "
110 PRINT : PRINT "     A      -
-TIRO--      ENTER"
120 PRINT : PRINT : PRINT TAB
6;"QUE VENCA O MELHOR"
130 FOR n=USR "a" TO USR "i"+7
140 READ d
150 POKE n,d
160 NEXT n
170 DATA 1,3,6,15,31,63,15,15
180 DATA 192,192,0,192,192,192
,0,135
190 DATA 15,15,15,15,15,12,12,
14
200 DATA 248,248,128,128,128,
192,192,224
210 DATA 0,0,0,0,0,0,126,0
220 DATA 3,3,0,3,3,3,0,113
230 DATA 128,192,96,240,248,
252,240,240
240 DATA 15,15,1,1,1,3,3,7
250 DATA 240,240,240,240,240,
48,48,112
260 LET y1=10: LET y2=10
270 LET b1=6: LET b2=6
280 LET s1=0: LET s2=0
290 RESTORE 330: FOR n=1 TO 8
300 READ d,p
310 SOUND d,p
320 NEXT n
330 DATA .1,7,.09,12,.1,7,.09,
12,.6,7,.45,2,.45,6,.5,0
340 PRINT #1;AT 0,0; FLASH 1;"
  QUALQUER TECLA PARA COMECAR
 "
350 LET p=254: GOSUB 570
360 IF i=191 THEN  GOTO 350
370 INK 4: BRIGHT 1: CLS
380 PRINT  INVERSE 1;"JOGADOR1
 0             0 JOGADOR2"
390 PRINT "BALAS :  6
   6  : BALAS"
400 PRINT #1;AT 0,0; INVERSE 1
;
410 GOSUB 600
420 LET p=63486: GOSUB 570
430 IF i=62 OR i=30 THEN
GOSUB 810
440 LET p=61438: GOSUB 570
450 IF i=62 OR i=30 THEN
GOSUB 840
460 LET p=64510: GOSUB 570
470 IF i=62 OR i=30 THEN
GOSUB 870
480 LET p=57342: GOSUB 570
490 IF i=62 OR i=30 THEN
GOSUB 900
500 LET p=49150: GOSUB 570
510 IF i=62 OR i=30 THEN
GOSUB 1020
520 LET p=65022: GOSUB 570
530 IF i=62 OR i=30 THEN
GOSUB 930
540 GOSUB 600
550 IF b1=0 AND b2=0 THEN
GOTO 1110
560 GOTO 420
570 LET i=IN p
580 IF i>191 THEN  LET i=i-64
590 RETURN
600 PRINT AT y1,1;CHR$ 144;
CHR$ 145
610 PRINT AT y1+1,1;CHR$ 146;
CHR$ 147
620 PRINT AT y2,29;CHR$ 149;
CHR$ 150
630 PRINT AT y2+1,29;CHR$ 151;
CHR$ 152
640 PRINT AT y1-1,1;"  "
650 PRINT AT y1+2,1;"  "
660 PRINT AT y2-1,29;"  "
670 PRINT AT y2+2,29;"  "
680 PRINT AT 0,9; PAPER 4; INK
9;s1
690 PRINT AT 0,22; PAPER 4;
INK 9;s2
700 PRINT AT 1,9;b1
710 PRINT AT 1,22;b2
720 RETURN
730 PRINT AT 10,10;"AAGH! ME A
CERTOU !"
740 RESTORE 780: FOR n=1 TO 11
750 READ d,p
760 SOUND d,p
770 NEXT n
780 DATA .5,2,.4,2,.2,2,.5,2,.
3,5,.2,4,.4,4,.2,2,.4,2,.2,1,.
5,2
790 PRINT AT 10,10;"
     "
800 RETURN
810 IF y1<4 THEN  RETURN
820 LET y1=y1-1
830 RETURN
840 IF y2<4 THEN  RETURN
850 LET y2=y2-1
860 RETURN
870 IF y1>18 THEN  RETURN
880 LET y1=y1+1
890 RETURN
900 IF y2>18 THEN  RETURN
910 LET y2=y2+1
920 RETURN
930 IF b1=0 THEN  RETURN
940 SOUND .01,4: SOUND .01,0
950 FOR n=3 TO 27
960 PRINT AT y1,n;" ";CHR$ 148
970 NEXT n
980 PRINT AT y1,27;"  "
990 IF y1=y2 OR y1=y2+1 THEN
LET s1=s1+1: GOSUB 730
1000 LET b1=b1-1
1010 RETURN
1020 IF b2=0 THEN  RETURN
1030 SOUND .01,0: SOUND .01,-10
1040 FOR n=27 TO 3 STEP -1
1050 PRINT AT y2,n;CHR$ 148;" "
1060 NEXT n
1070 PRINT AT y2,3;" "
1080 IF y2=y1 OR y2=y1+1 THEN
LET s2=s2+1: GOSUB 730
1090 LET b2=b2-1
1100 RETURN
1110 IF s1>s2 THEN  PRINT AT 10
,5; FLASH 1;" JOGADOR 1 VENCEU
!"
1120 IF s2>s1 THEN  PRINT AT 10
,5; FLASH 1;" JOGADOR 2 VENCEU
!"
1130 IF s1=s2 THEN  PRINT AT 10
,10; FLASH 1;" EMPATE ! "
1140 GOTO 260
```

```
10 GOTO 170
15 S1=S1-1:LINE(254,Y1+5)-(20,Y
1+5),15:LINE(254,Y1+5)-(20,Y1+5
),1
20 IFY1>Y2-5ANDY1<Y2+13THENP1=P
1+1:PUTSPRITE2,(X2,Y2),6,2
25 FORT=0TO50:NEXT:RETURN
30 S2=S2-1:LINE(0,Y2+5)-(235,Y2
+5),15:LINE(0,Y2+5)-(235,Y2+5),
1
35 IFY2>Y1-5ANDY2<Y1+13THENP2=P
2+1:PUTSPRITE1,(X1,Y1),6,2
40 FORT=0TO50:NEXT:RETURN
45 CLS:COLOR 15,2,2
50 LOCATE12,2:PRINT"DUELO ESTEL
AR"
55 PRINT:PRINT"ganhe pontos ace
rtando o seu oponente"
60 PRINTTAB(5)"cada jogador tem
 seis balas"
65 PRINT:PRINT:PRINT"jogador 1"
;SPC(19);"jogador 2":PRINT
70 PRINTTAB(3)"E  -------   cim
a  -------  I"
75 PRINTTAB(3)"D  -------  baix
o  -------  J"
80 PRINTTAB(3)"F  -------   fog
o  -------  H"
85 LOCATE10,20:PRINT"<qualquer
tecla>"
90 IF INKEY$="" THEN 90
95 COLOR 15,1,1:SCREEN 2,1
100 SPRITE$(0)=A$
105 SPRITE$(1)=B$
110 SPRITE$(2)=C$
115 IF S1=0 OR S2=0 THEN 225
120 OUT C,(INP(C)ANDH)OR3
125 I=INP(B)
130 IF(IAND2)=0ANDY1<176THEN Y1
=Y1+2
135 IF(IAND4)=0ANDY1>1THEN Y1=Y
1-2
140 IF(IAND8)=0 THEN GOSUB15
145 IF(IAND32)=0 THEN GOSUB30
150 IF(IAND64)=0ANDY2>1THEN Y2=
Y2-2
155 IF(IAND128)=0ANDY2<176THEN
Y2=Y2+2
160 PUTSPRITE1,(X1,Y1),15,0:PUT
SPRITE2,(X2,Y2),15,1
```

```
165 GOTO 115
170 S1=6:S2=6:P1=0:P2=0:C=170:H
=240
175 X1=5:X2=230:Y1=96:Y2=96:B=1
69
180 FOR K=1 TO 8
185 READ P,Q,R
190 A$=A$+CHR$(P)
195 B$=B$+CHR$(Q)
200 C$=C$+CHR$(R)
205 NEXT K
210 DATA 48,12,129,48,12,66,255
,255,36,112,14,24
215 DATA 48,12,24,40,20,36,72,1
8,66,144,9,129
220 GOTO 45
225 SCREEN0:CLS
230 CLS:PRINT"FIM DE JOGO":PRIN
T
235 PRINT:PRINT"jogador 1 = ";P
1" pontos"
240 PRINT"jogador 2 = ";P2;" po
ntos"
245 LOCATE8,20:PRINT"JOGA DE NO
VO (S/N) ?"
250 I$=INKEY$:IFI$="S"THEN RUN
255 IF I$<>"N"THEN 250
260 CLS:COLOR15,4,4:END
```

```
10 P = 36800:Q = 32:S1 = 6:S2 =
 6:P1 = 0:P2 = 0:X1 = 1
15 X2 = 271:Y1 = 96:Y2 = 96:H1
= 1:H2 = 271:V1 = 96:V2 = 96
20  GOTO 85
25 S1 = S1 - 1
30  HPLOT 20,Y1 TO 279,Y1: HCOL
OR= 0
35  HPLOT 20,Y1 TO 279,Y1: HCOL
OR= 3
40  IF Y1 > Y2 - 3 AND Y1 < Y2
+ 6 THEN  GOTO 75
45  RETURN
50 S2 = S2 - 1
55  HPLOT 260,Y2 TO 0,Y2: HCOLO
R= 0
60  HPLOT 260,Y2 TO 0,Y2: HCOLO
R= 3
65  IF Y2 > Y1 - 3 AND Y2 < Y1
+ 6 THEN  GOTO 80
70  RETURN
75 P1 = P1 + 1: PRINT  CHR$ (7)
: RETURN
80 P2 = P2 + 1: PRINT  CHR$ (7)
: RETURN
85  HOME :E = 35000: HIMEM: E
90 F =  INT (E / 256): POKE 232
,E - F * 256: POKE 233,F
95  FOR I = E TO E + 41 + 2 * 3
2
100  READ A: POKE I,A
105  NEXT
110  SCALE= 1: ROT= 0
115  DATA    20 ,0 ,42 ,0 ,74 ,
0 ,106 ,0 ,138 ,0 ,170 ,0 ,202
,0 ,234 ,0 ,10 ,1 ,42 ,1 ,74 ,1
 ,106 ,1 ,138 ,1 ,170 ,1 ,202 ,
1 ,234 ,1 ,10 ,2 ,42 ,2 ,74 ,2
,106 ,2 ,138,2
120  DATA   0 ,72 ,109 ,73 ,218
 ,219 ,255 ,42 ,45 ,45 ,45 ,213
 ,219 ,59 ,191 ,9 ,109 ,73 ,218
 ,27 ,31 ,159 ,105 ,105 ,137 ,2
19 ,27 ,223 ,6 ,0 ,0 ,0
125  DATA   0 ,72 ,9 ,109 ,209
,27 ,255 ,155 ,45 ,45 ,45 ,173
,27 ,63 ,223 ,83 ,73 ,109 ,209
,27 ,31 ,223 ,74 ,105 ,105 ,26
,223 ,223 ,19 ,0 ,0 ,0
130  REM  SUBROTINA DE LEITURA
135  REM     DO TECLADO
140  FOR K = 36810 TO 36815
145  READ A: POKE K,A: NEXT
150  DATA  64,32,16,8,4,2
155  FOR K = 36900 TO 36935
160  READ A: POKE K,A: NEXT
165  DATA    162,6,169,0,157,19
1,143,202,224,0,208,248,162,0,1
89,202,143,141
170  DATA   0,192,169,32,45,16,
192,240,3,157,192,143,232,224,6
,208,235,96
175  VTAB (4): INVERSE : PRINT
 TAB( 39)" ": NORMAL
180  PRINT  TAB( 15)" D U E L O
"
185  INVERSE : PRINT  TAB( 39)"
 ": NORMAL
190  PRINT : PRINT " GANHE PONT
OS ACERTANDO O SEU OPONENTE"
195  PRINT  TAB( 6)"CADA JOGADO
R TEM SEIS BALAS"
200  PRINT : PRINT : PRINT "JOG
ADOR 1 ------------------ JOGA
DOR 2": PRINT
205  PRINT "   1   ----------CI
MA---------   0"
210  PRINT "   Q   ----------FO
GO---------   P"
215  PRINT "   A   ---------BAI
XO---------   :"
220  PRINT : PRINT : PRINT : PR
INT
225  HTAB (13): INVERSE : PRINT
 "QUALQUER TECLA": NORMAL
230  GET Z$: IF Z$ = "" THEN 23
0
235  HGR2 : HCOLOR= 3
240  DRAW 1 AT X1,Y1: DRAW 2 AT
 X2,Y2
245  XDRAW 1 AT H1,V1: XDRAW 2
AT H2,V2
250 V1 = Y1:V2 = Y2
255  DRAW 1 AT X1,Y1: DRAW 2 AT
 X2,Y2
260  CALL 36900
265  IF S1 = 0 OR S2 = 0 THEN
GOTO 305
270  IF  PEEK (P) = Q AND Y2 <
181 THEN Y2 = Y2 + 3
275  IF  PEEK (P + 1) = Q THEN
 GOSUB 50
280  IF  PEEK (P + 2) = Q AND Y
2 > 10 THEN Y2 = Y2 - 3
285  IF  PEEK (P + 3) = Q AND Y
1 > 10 THEN Y1 = Y1 - 3
290  IF  PEEK (P + 4) = Q THEN
 GOSUB 25
295  IF  PEEK (P + 5) = Q AND Y
1 < 181 THEN Y1 = Y1 + 3
300  GOTO 245
305  TEXT : HOME
310  PRINT "FIM DE JOGO": PRINT
 : PRINT
315  PRINT "JOGADOR 1 = ";P1;"
PONTOS"
320  PRINT "JOGADOR 2 = ";P2;"
PONTOS"
325  PRINT : PRINT : INPUT "JOG
A DE NOVO (S/N) ?";A$
330  IF A$ = "S" THEN  RUN
335  END
```

OS SEGREDOS DO TRS-80 (3)

- MAIS USOS PARA VTRANSF
- CÓPIA DA TELA EM FITA OU DISCO
- COMO RECUPERAR UMA TELA GRAVADA

Como vimos em artigos anteriores, podemos utilizar vários "truques" para explorar melhor os recursos de vídeo do TRS-80. Aprenderemos agora a gravar uma tela em fita ou disco.

A sub-rotina **VTRANSF**, apresentada no segundo artigo desta série, será usada aqui por um programa que copia integralmente o conteúdo da tela de vídeo, armazenando-o, depois, em um arquivo de fita ou disco. Um outro programa permitirá a realização da operação inversa, ou seja, colocar novamente na tela um desenho ou texto armazenado em fita ou disco.

O método utilizado para isso é semelhante ao que vimos no artigo anterior, quando tratamos da impressão, linha por linha, do conteúdo da tela. A diferença é que, em vez de imprimirmos a variável **V$**, simplesmente a enviamos para fita ou disco.

ARMAZENAGEM EM FITA

Listamos, a seguir, a versão para fita. Depois de digitá-la, acrescente a sub-rotina 1000, apresentada no artigo anterior.

```
10 CLEAR 500
20 CLS : PRINT "PREPARE O GRAV
ADOR"
25 PRINT "E PRESSIONE <ENTER> "
30 C%=64 : A$=CHR$(34)
35 IF INKEY$="" THEN 35
40 FOR V%=0 TO 960 STEP 64
50 GOSUB 1000:PRINT #-1,A$;V$;
A$
60 NEXT V%
70 END
```

A linha 30 determina o comprimento da linha de vídeo a ser copiada (54 posições). Ela se encarrega, também, de definir a variável **A$**, que conterá o caractere "aspas". Este será usado na linha 50, para englobar a variável **V$**. A linha 35 espera que o usuário prepare o gravador e pressione a tecla <**ENTER**>. Só então o programa continua. O laço que vai da linha 40 à linha 60 copia as dezesseis linhas da tela, sucessivamente, na variável **V$**, usando a função **VARPTR**, e, em seguida, envia para o gravador 1.

É muito importante colocar todo o *string* **V$** entre aspas, pois, se houver alguma vírgula ou ponto e vírgula na tela, a transmissão para a fita será truncada, provocando um erro de leitura, posteriormente.

ARMAZENAGEM EM DISCO

A versão para disco é mais simples:

```
10 CLEAR 500
20 OPEN "O",1,"TELA1/VID"
30 C%=64
40 FOR V%=0 TO 960 STEP 64
50 GOSUB 1000 : PRINT #1,V$
60 NEXT V%
70 CLOSE 1:END
```

A linha 20 abre um arquivo de acesso seqüencial para saída, chamado TELA1/VID (poderia ser qualquer nome). As linhas 30 a 60 funcionam como na versão para fita, só que não são necessárias as aspas em torno de **V$**. A linha 70 fecha o arquivo criado.

LEITURA EM FITA

O programa que lê o conteúdo de tela no arquivo, trazendo-o de volta ao vídeo, obedece à mesma seqüência:

```
10 CLEAR 500
20 CLS : PRINT "PREPARE O GRAV
ADOR"
25 PRINT "E PRESSIONE <ENTER> "
30 C%=64
35 IF INKEY$="" THEN 35
40 FOR V%=0 TO 960 STEP 64
50 GOSUB 1000 : INPUT #-1,X$
55 V$=LEFT$(X$,C%)
60 NEXT V%
70 END
```

Observe que a sub-rotina **VTRANSF** também serve para o programa de leitura, sem modificações, já que sua função consiste em determinar a localização e tamanho do *string* **V$** na memória. Assim, depois de definida pela linha 55, a variável **V$** é automaticamente armazenada na memória de vídeo. Sua exibição na tela também é instantânea.

A função **LEFT$** toma apenas os 64 primeiros caracteres de **X$**, uma vez que, na gravação, um caractere ASCII 13 (*linefeed*) foi adicionado ao final de **V$**. Esse caractere prejudicaria toda a tela se não fosse retirado.

LEITURA EM DISCO

A versão do programa de leitura em disco é a seguinte:

```
10 CLEAR 500
20 OPEN "I",1,"TELA1/VID"
30 C%=64
40 FOR V%=0 TO 960 STEP 64
50 GOSUB 1000:LINE INPUT #1,X$
55 LSET V$=X$
60 NEXT V%
70 CLOSE 1:END
```

O **LSET** da linha 55 existe apenas no BASIC para disco. Tem a mesma função do **LEFT$** da versão para fita mas, em termos operativos, é mais simples que esse comando.

APLICAÇÕES

Existem muitas aplicações para os programas listados neste artigo. Uma das mais comuns é na montagem das telas de instruções de um jogo. Estas, quando mais elaboradas, podem incluir gráficos, juntamente com textos explicativos — por exemplo, a reprodução do desenho de cada tipo de nave inimiga e o número de pontos que o jogador ganha se abatê-las. Em vez de colocar todos esses dados dentro de uma mesma listagem, tornando-a longa e complicada, use os programas de transferência para jogar as figuras e o texto diretamente da fita ou disco para o vídeo do microcomputador, gastando apenas 64 bytes de memória RAM!

Os programas educativos constituem uma outra área importante de aplicação. As telas podem compor, por exemplo, as páginas de um "livro eletrônico". Armazenadas em disco, serão "folheadas" pelo aluno, que copiará aquelas que julgar necessário.

AVALANCHE: ACERTO DAS VARIÁVEIS

- O AVANÇO DA MARÉ
- GAIVOTAS, NUVENS E VENTO
- PEDRAS E COBRAS
- SITUAÇÃO DO PERSONAGEM

Uma das tarefas mais difíceis na programação de jogos em código de máquina é garantir a sincronia dos diversos eventos. Aqui estão as rotinas que cuidam disso.

Sempre que vamos recomeçar o jogo, precisamos acertar o escore e, também, estabelecer os parâmetros que influenciam o comportamento de outras sub-rotinas. Temos que esvaziar a maré que ameaçava Willie, definir a direção do vento, colocar Willie de volta ao pé do monte e, para que tudo ocorra no momento adequado, acertar os diversos laços controladores de tempo.

A rotina que se segue acerta o valor das diversas variáveis para que Willie possa se dedicar à árdua tarefa de recuperar seu lanche roubado.

```
10  REM org 58606
20  REM dth ld a,6
30  REM ld (57353),a
40  REM ld hl,736
50  REM ld (57354),hl
60  REM ld hl,130
70  REM ld (57345),hl
80  REM ld a,3
90  REM ld (57347),a
100 REM ld a,0
110 REM ld (57348),a
120 REM ld a,2
130 REM ld (57349),a
140 REM ld hl,449
150 REM ld (57332),hl
160 REM ld hl,0
170 REM ld (57334),hl
```

```
180 REM ld a,0
190 REM ld (57336),a
200 REM ld hl,223
210 REM ld (57356),hl
220 REM ld a,0
230 REM ld b,5
240 REM ld (57350),a
250 REM add a,b
260 REM ld (57351),a
270 REM add a,b
280 REM ld (57352),a
```

Essa rotina não é chamada apenas no início do jogo, mas, também, quando ele recomeça, depois de Willie encontrar a morte dentro de um buraco, afogado ou embaixo de alguma pedra.

O MAR

Colocando todas as variáveis juntas na memória do microcomputador, podemos verificar o status do jogo a qualquer instante, o que facilita muito a tarefa de depuração de erros.

Variáveis de um byte são modificadas através do acumulador, que tem oito bits, enquanto o par de registros HL se encarrega das variáveis de dois bytes, mesmo que o seu conteúdo no momento caiba em apenas um byte. Isso ocorre porque o byte mais significativo também deve ser modificado.

O endereço 57353 contém o atraso do movimento da maré. Colocamos ali o número 6, dando a Willie uma chance razoável de escalar a montanha antes de se afogar. Depois, esse valor poderá ser modificado para acelerar o avanço da maré, o que tornará o jogo mais difícil e mais emocionante.

O mar deve ser desenhado a partir da base da tela, a cada nova etapa do jogo. A posição da tela correspondente ao canto superior esquerdo do mar é colocada nos endereços 57354 e 57355. O número 736, valor inicial dessa variável, equivale ao endereço da extremidade inferior esquerda da tela.

A NUVEM

Na versão do jogo para o Spectrum, há uma nuvem cruzando o céu. O endereço 57345 contém sua posição de impressão na tela. O valor inicial dessa variável é 130.

Mas o programa utiliza outros dados para cuidar do movimento da nuvem. Se não provocarmos um atraso, ela atravessará o céu tão rapidamente quanto um avião a jato. Para evitar isso, colocamos um contador de atraso na posição 57347. Seu valor inicial é 3.

Precisamos também saber em que di-

reção sopra o vento, para que a nuvem se mova de acordo. A informação fica armazenada em 57348. Quando esse byte vale zero, a nuvem se movimenta para a direita; quando vale 1, para a esquerda. No programa, essa variável tem valor inicial zero, movendo, portanto, a nuvem para a direita.

VIDA OU MORTE

Um atraso para o movimento das gaivotas é colocado em 57349, com valor inicial 2. A posição que Willie ocupa na tela fica em 57352. O programa coloca ali o valor 449, correspondente ao canto inferior esquerdo do perfil da encosta da montanha.

Uma outra variável informa ao programa se nosso herói está parado, correndo ou saltando. Por motivos que conheceremos mais adiante, utilizaremos dois bytes, 57334 e 57335. Como Willie começa o jogo em pé, parado no sopé da montanha, o programa coloca zero naquelas posições.

A situação atual do jogo pode ser monitorada pela variável que fica em 57336. Se esse byte for 0, Willie vai bem, obrigado. Se for 1, ele acaba de recuperar parte do seu lanche e as rotinas do próximo nível de dificuldade vão ser chamadas. Se for 2, Willie morreu! Como no início do jogo nosso personagem está vivo, o programa coloca 0 nessa posição.

PEDRAS E COBRAS

A variável que controla a posição da pedra fica em 57356. O programa coloca ali seu valor inicial, 223, que corresponde ao topo da montanha, no canto superior direito.

As três cobras têm língua que se mexe. Como não é desejável que as línguas façam o mesmo movimento, precisamos provocar uma defasagem em seu ritmo. Para isso, colocam-se variáveis de atraso diferentes para cada língua, nos endereços 57350, 57351 e 57352. O acumulador recebe o valor zero, e o registro B, o número 5. Na primeira variável de atraso coloca-se, então, zero. Soma-se o conteúdo de B ao acumulador e o resultado, 5, é colocado no segundo atraso. Mais uma vez 5 é somado ao acumulador e o resultado, 10, vai para o terceiro atraso.

T

O programa em Assembly listado a seguir estabelece o valor inicial de uma série de variáveis que são usadas para controlar o jogo.

```
10   ORG 19447
20   NLV LDA #6
30   STA 18246
40   LDX #7424
50   STX 18247
60   LDX #5088
70   STX 18249
80   CLR 18251
90   CLR 18252
100   LDX #3070
110   STX 18253
120   CLR 18255
130   LDA 5
140   STA 18256
150   LDA #10
160   STA 18257
170   RTS
```

Essa rotina, que recebeu o rótulo **NLV**, é utilizada no início do jogo e, também, quando Willie morre.

O MAR

Uma das vantagens de se colocar todas as variáveis em um mesmo local da memória é que podemos verificar facilmente o status do jogo quando estamos depurando os erros.

O valor de variáveis de um byte é estabelecido com a ajuda do acumulador, que tem oito bits, enquanto o registro X, de dezesseis bits, se encarrega das variáveis de dois bytes, mesmo que o seu conteúdo, no momento, caiba em um único byte. Isso ocorre porque o byte mais significativo também deve ser modificado.

O endereço 18246 contém o atraso do movimento da maré. O programa coloca o valor 6 nessa posição para que Willie tenha uma chance razoável de subir a montanha antes de se afogar. Depois, poderemos mudar esse valor para acelerar o avanço da maré, tornando o jogo mais difícil e emocionante.

O mar deve ser desenhado a partir da base da tela, a cada nova etapa do jogo. A posição da memória de vídeo correspondente ao canto superior esquerdo do mar é colocada nos bytes 18247 e 18248. O programa guarda ali o valor inicial 7424, que corresponde ao canto inferior esquerdo da tela.

VIDA OU MORTE

A posição de Willie na tela fica armazenada em 18249. O programa coloca nesse byte o valor inicial 5088, correspondente ao canto inferior esquerdo do perfil da encosta da montanha, de

onde nosso herói deve partir em busca do lanche perdido

Uma outra variável, que fica em 18251, verifica se Willie está parado, andando ou pulando. O programa coloca valor zero nessa posição já que no início do jogo o personagem está parado, ao pé da montanha.

A situação atual do jogo pode ser monitorizada pela variável que fica em 18252. Se esse byte for 0, Willie vai bem, obrigado. Se for 1, ele acaba de recuperar parte do seu lanche e as rotinas do próximo nível de dificuldade vão ser chamadas. Se for 2, Willie morreu! Como no início do jogo nosso personagem está vivo, o programa coloca 0 nessa posição.

PEDRAS E COBRAS

A variável que controla a posição da pedra fica em 18253. O programa coloca ali seu valor inicial, 3070, que corresponde ao topo da montanha, no canto superior direito.

As três cobras têm língua que se mexe. Como não é desejável que as línguas façam o mesmo movimento, precisamos provocar uma defasagem em seu ritmo. Para isso, colocam-se variáveis de atraso diferentes para cada língua, nos endereços 18255, 18256 e 18257. A primeira variável de atraso é limpa com **CLR**, tornando-se zero. Os números 5 e 10 são colocados, respectivamente, na segunda e na terceira variáveis de atraso, com o auxílio do acumulador.

A rotina listada a seguir ajusta uma série de variáveis com os valores que elas precisam conter quando Willie inicia sua arriscada missão.

```
10    org -11645
20    mrt ld a,6
30    ld (-5213),a
40    ld hl,736
50    ld (-5212),hl
60    ld hl,130
70    ld (-5210),hl
80    ld a,3
90    ld (-5208),a
100    ld a,0
110    ld (-5207),a
120    ld a,2
130    ld (-5206),a
140    ld hl,481
150    ld (-5205),hl
160    ld hl,0
170    ld (-5203),hl
180    ld a,0
190    ld (-5201),a
200    ld hl,255
210    ld (-5200),hl
220    ld a,0
230    ld b,5
240    ld (-5198),a
250    add a,b
260    ld (-5197),a
270    add a,b
280    ld (-5196),a
```

Essa rotina, rotulada **mrt**, não é chamada apenas no início do jogo, mas, também, após a morte de Willie, quando as variáveis são reajustadas para uma nova partida.

O MAR

Como reunimos todas as variáveis em um mesmo lugar da memória, podemos verificar o status do jogo a qualquer momento, quando precisamos corrigi-lo.

As variáveis de um byte são ajustadas pelo acumulador de oito bits, enquanto o par HL, que tem dezesseis bits, se encarrega das variáveis de dois bytes, mesmo que o valor nelas armazenado, em determinado estágio, caiba em um só byte. Isso ocorre porque os bits mais significativos da variável também devem ser ajustados.

A posição de memória −5213 contém o atraso do mar. Ela é carregada com o valor 6, para que Willie tenha

uma razoável chance de escalar a montanha antes de se afogar. Depois, poderemos modificar esse valor para acelerar o avanço da maré, tornando o jogo mais difícil e emocionante.

No início de cada tentativa de Willie, o mar deve estar bem baixo, na parte inferior do cenário. A posição do canto esquerdo da superfície do mar é armazenada em −5212 e −5211.

O número 736 é carregado nessa posição, que corresponde ao canto inferior esquerdo da tela.

A NUVEM

Na versão de *Avalanche* para o MSX, a nuvem move-se no céu acima da montanha. Os endereços −5210 e −5209, que contêm sua posição na tela, são carregados com 130, que corresponde ao ponto onde ela surge.

Mas o programa utiliza outros dados para cuidar do movimento da nuvem. Se não provocarmos um atraso, ela atravessará o céu como um jato.

Para ajustar a variável de atraso, carrega-se o endereço em que está armazenada, −52080, com o valor 3.

Precisamos também definir a direção em que a nuvem se move. Esta informação é armazenada em −5207. Se colocarmos 0 nesse endereço, ela se dirigirá para a direita; se colocarmos 1, para a esquerda. Como a rotina será inicializada tendo 0 nessa posição, a nuvem irá para a direita.

VIDA OU MORTE

O atraso do movimento da gaivota é armazenado no endereço −5206, sendo, inicialmente, ajustado com o valor 2. A posição de Willie na tela é determinada pelo conteúdo dos endereços −5205 e −5204, que são carregados com 481, valor correspondente ao lado esquerdo da base da encosta.

Uma outra variável controla a movimentação de Willie, se ele estiver de pé, correndo ou pulando. Por razões que veremos depois, ela é armazenada em dois bytes, −5302 e −5202. Como, no início do jogo, Willie está de pé, esses endereços são ajustados em 0.

A condição geral do jogo é monitorada pela variável, que fica no endereço −5201. Vamos chamá-la de variável morte. Se seu valor for 0, está tudo bem com Willie. Se for 1, nosso personagem alcançou seu prêmio e a próxima tela do jogo deverá ser chamada. Um valor 2 significa que Willie está morto! Como o jogo começa com Willie vivo, esse byte é carregado com 0.

PEDRAS E COBRAS

A variável que controla a posição da pedra é armazenada nos endereços −5200 e −5199. Essa variável é ajustada com o valor 255, que corresponde ao lado direito do topo da encosta.

As três cobras têm língua que se mexe. Como não é desejável que as línguas tenham o mesmo movimento, devemos provocar uma defasagem em seu ritmo. Para isso, carrega-se o acumulador A com o valor 0 e o registro B com a defasagem 5. O valor 0 é armazenado na variável de atraso da primeira cobra (−5198). A defasagem 5 é adicionada ao valor de A (0) pela instrução **add a,b** e o resultado, 5, vai para a variável de atraso da segunda cobra (−5197). Repete-se a operação e o resultado, 10, é armazenado na variável de atraso da terceira cobra (−5196).

MOUSE MECÂNICO E MOUSE ÓPTICO

O *mouse* é um periférico cada vez mais procurado por usuários de microcomputadores pessoais ou profissionais. Conheça-o de perto e descubra suas várias utilidades.

A busca por novas formas de interação entre o usuário e o computador tem levado ao desenvolvimento de diversos tipos de periféricos, como o joystick, a caneta óptica e o tablete de digitalização, que têm por finalidade facilitar o deslocamento de um cursor, de texto ou gráfico, sobre a tela.

Um dos periféricos dessa família é o *mouse*, ou camundongo, dispositivo que vem sendo cada vez mais usado em microcomputadores de todos os tipos.

O mouse recebeu essa denominação em virtude da semelhança física com um ratinho: é uma pequena caixa achatada, de cantos arredondados, com um a três botões planos em sua superfície dorsal e um fio (o "rabo") que sai da parte traseira, conectado ao computador.

Para usá-lo, repousa-se uma das mãos sobre a caixinha, ao mesmo tempo apoiando um ou mais dedos sobre os botões. Para efetuar algum movimento do cursor sobre a tela de vídeo, basta deslocá-lo levemente de um lado para outro sobre uma superfície dura e plana. Com um mínimo de prática, o usuário aprende a coordenar o mouse com o que acontece na tela, conseguindo grande agilidade e flexibilidade em inúmeras funções fixadas por um software especial — entre elas, selecionar itens em um menu, desenhar sobre a tela, apontar um cursor ou uma mira etc.

Os botões da parte de cima do mouse têm um papel idêntico ao do botão de disparo de um joystick, sendo usados, normalmente, para selecionar opções.

DA ORIGEM À POPULARIZAÇÃO

O mouse nasceu em um centro de pesquisas norte-americano, que estava desenvolvendo uma nova linguagem, chamada *small-talk*, e precisava de um terminal de vídeo especial para implementar os novos conceitos de interatividade que seus criadores imaginaram. O periférico que eles criaram era uma espécie de *trackball* às avessas: em vez de deslocar com a palma da mão uma esfera presa ao teclado, o usuário deslocava a esfera sobre uma superfície.

Inicialmente, o uso desse dispositivo foi bastante restrito, já que era disponível apenas para minicomputadores muito caros. Porém, a partir do aparecimento dos microcomputadores das linhas Lisa e Macintosh (fabricados pela Apple Computer nos EUA), que basearam seus sistemas operacionais na interação com o mouse, ele passou a ser fabricado também para outras linhas de microcomputadores. Em pouco tempo, tornou-se muito conhecido.

No Brasil, o mouse está disponível para microcomputadores das linhas ZX Spectrum (TK-90X e TK-95), Apple II e MSX. No exterior podem ser encontrados modelos para micros de outras linhas. Em um artigo futuro, ensinaremos algumas técnicas em BASIC para o mouse.

Embora um mouse tenha sempre a mesma aparência externa, com poucas variações de um modelo para outro, existem, na realidade, dois tipos fundamentalmente diferentes quanto ao mecanismo de funcionamento: o mouse mecânico e o mouse óptico.

O primeiro tem uma esfera de plástico ou metal maciço em sua "barriga" (ou seja, no lado voltado para a mesa). Quando o dispositivo é deslocado, a esfera gira ao redor de seu centro, acompanhando o movimento. Sensores colocados em sua superfície, dentro da caixinha, enviam ao computador os ângulos de rotação verificados em duas dimensões e um software especial, previamente carregado na memória do computador, transforma esses ângulos em coordenadas X e Y, que são usadas para posicionar o cursor na tela.

O mouse óptico funciona de acordo com um princípio totalmente diferente: um diodo luminescente (*LED*), localizado na parte inferior da caixinha, lança um feixe de luz em direção à superfície sobre a qual será deslocado o mouse. Em outro orifício, um sensor luminoso capta as variações da luz refletida pela superfície.

Ao contrário do mouse mecânico, que pode ser deslocado sobre qualquer superfície plana e não muito lisa, o mouse óptico requer uma superfície especial. Esta consiste de um tablete todo marcado com um quadriculado em cor azul ou preta. Ao se deslocar o mouse sobre ela, as linhas do quadriculado provocam uma alteração na reflexão do feixe de luz vermelha emitido pelo diodo, o que é sentido pelo software especial como uma movimentação em algum sentido. Essa informação, por sua vez, é transmitida ao computador e convertida em coordenadas X e Y.

Cada tipo de mouse apresenta vantagens e desvantagens. O modelo mecânico é mais frágil, quebrando-se facilmente, e menos sensível do que o óptico. Porém, pode ser usado sobre qualquer tipo de superfície. Já o mouse óptico requer uma superfície especial, limitada em tamanho, e apresenta ainda a desvantagem de necessitar de uma fonte de alimentação — dependendo do modelo, ela precisa ser separada da fonte de alimentação do computador.

A maioria dos periféricos de tipo mouse é conectada ao micro através da porta de joystick analógico ou de uma porta serial do tipo RS-232.

Antes de adquirir um mouse, verifique suas características para se certificar de que poderá compatibilizá-lo com seu computador.

APLICAÇÕES

O mouse é um periférico muito divertido e fácil de usar, tanto em programas para jogos de qualquer tipo, quanto em aplicações mais "sérias".

A princípio, toda aplicação que permite a utilização de canetas ópticas ou joysticks também se presta para o uso do mouse. Existem, porém, alguns programas comercialmente disponíveis, que são feitos exclusivamente para o mouse. Incluem-se, entre eles, programas de desenho sobre a tela, de seleção rápida de menus etc.

E não se esqueça: com alguns comandos simples em BASIC, não será difícil desenvolver os seus próprios programas para o irrequieto mouse.

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craft II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Craft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:


Sinclair ZX-81


TRS-80


TK-2000


MSX


Spectrum


TRS-Color


Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NO PRÓXIMO NÚMERO

CÓDIGO DE MÁQUINA

Contagem de pontos em *Avalanche*. Exibição do placar. Número de vidas. Troca de telas. Níveis de dificuldade.

PROGRAMAÇÃO BASIC

Geração de acordes no MSX. Melodias simultâneas. A instrução **SOUND**. Tabela de conversão de notas.

PROGRAMAÇÃO DE JOGOS

Jogos de guerra no tabuleiro e no computador. *Capa e Espada*: organização do jogo e criação dos blocos gráficos.